République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieure et de la Recherche Scientifique

Université Mouloud Mammeri de Tizi-Ouzou

Faculté du Génie Electrique et Informatique

Département d'Informatique

Mémoire de fin d'études

En vue de l'obtention du diplôme Master

Domaine : Mathématiques et informatique

Filière : Informatique

Spécialité : Conduite de projets informatique

Thème

« La classification multi label hiérarchique à base d'analyse de concepts formels »

Soutenu le :05/11/2020

Présente par :

Mlle ROUIBAH Faiza

Mlle BOUDIA Djouher

Encadré par :

Mr RADJA Hakim

Membres du jury :

Mr CHEBOUBA Lokmane

Mr SAIDANI Fayçal

Année Universitaire : 2019/2020

Remerciements

Nous tenons à remercier chaleureusement notre promoteur Mr RADJA pour son suivi et ses orientations et lui exprimer notre reconnaissance pour le temps qu'il nous a consacré.

Nous remercions également les membres du jury d'avoir accepté de juger notre travail.
Nous exprimons notre profonde gratitude à tous les enseignants du département informatique ainsi qu'aux enseignants de la spécialité CPI.

Nos remerciements vont enfin à nos deux familles ainsi qu'à toute personne ayant contribué de près ou de loin à l'élaboration de ce modeste travail.

Dédicace

« Je dédie ce travail à mes chers parents, je ne les remercierai jamais assez pour tous leurs sacrifices, à mes chers frères et sœurs et à tous mes proches.»

Djouher

« Je dédie ce travail à mes chers parents qui m'ont soutenu jusqu'à à la fin, à mes chers frères et sœurs et à tous ceux qui me sont proches. »

Faiza

SOMMAIRE

Table des figures

Table des tableaux

Introduction générale

Dans la classification multi-label, chaque instance peut appartenir à une ou plusieurs classes simultanément. Très souvent dans le cas d'application réelle, les classes ne sont pas indépendantes les unes des autres.

Chaque objet pouvant avoir plusieurs labels (classes), il semble pertinent de commencer par déterminer s'il existe un lien entre ces labels, cela en étudiant les corrélations entre ces classes. Cette première étude permet de voir à quel point les classes sont structurées pour ensuite exploiter cette information pour, entre autres, choisir une méthode de classification adaptée, prenant en compte ou non le lien entre les labels ou bien corriger quelques résultats d'une première classification.

La classification multi-labels hiérarchique traite des problèmes où les classes sont organisées sous forme d'une hiérarchie. Ce domaine de recherche n'a pas eu une grande attention, et pourtant on retrouve ce problème dans beaucoup d'applications importantes. En effet, beaucoup de collections de textes sont organisées sous forme de hiérarchies, y compris :

- Les bibliothèques numériques et les référentiels web tels que DMOZ, Wikipédia, Yahoo, Looksmart, EUROVOC et Reuters.

- Le domaine de la bio-informatique, par exemple OHSUMED, FunCat et Gene Ontology.En marketing, en évalue la relation entre l'exposition à une publicité télévisuelle et l'acte d'achat.

- La reconnaissance d'images.

Cette organisation hiérarchique est essentielle parce qu'elle permet à l'utilisateur de se concentrer sur le niveau approprié de détails. A partir de là, nous pouvons légitimement penser à l'analyse de concepts formels et aux treillis de Galois. En effet, en analyse de concepts formels, les connaissances induites (appelées concepts formels) à partir d'un contexte formel sont hiérarchisées et représentées sous la forme d'un treillis de Galois.

l'ACF consiste à induire des paires de sous-ensembles $\langle\{objets\},\{propriétés\}\rangle$, appelées concepts formels, à partir d'une relation binaire entre un ensemble d'objets et un

ensemble de propriétés. Elle a été utilisée dans divers domaines : psychologie, sociologie, biologie, médecine, linguistique, mathématiques, informatique, etc...

Les connaissances induites appelées concepts formels sont hiérarchisées et représentées sous la forme d'un treillis de Galois. Les treillis de Galois constituent un moyen efficace permettant d'avoir une représentation exhaustive de la réalité étudiée (contexte formel).

L'objectif principal de notre travail consiste à présenter une approche de la classification basée sur l'analyse de concepts formels qui permet d'obtenir une hiérarchie entre les concepts. Cette dernière consiste à construire des modèles appelés classifieurs à partir des données permettant de prédire les classes des futures données.

Notre mémoire s'articule sur les quatre (4) chapitres suivants :

Chapitre 1 : La classification multi label : Dans ce chapitre nous parlerons d'une manière générale de la classification multi label et ses différentes méthodes. Nous avons utilisé les mesures de performance Pour évaluer ses algorithmes.

Chapitre 2 : La classification multi label hiérarchique : Dans ce chapitre nous avons présenté quelques notions relatives à la classification multi label hiérarchique, et ses différentes méthodes

Chapitre 3 : Analyse de concepts formels : Dans ce chapitre nous avons présenté quelques notions relatives à l'analyse de concepts formels et les algorithmes de construction de treillis de Galois.

Chapitre 4 : Apport : Nous avons présenté les phases de notre approche de classification multi label hiérarchique basée sur l'analyse de concepts formels.

Chapitre 1

La classification Multi label

1. INTRODUCTION :

Depuis l'aube des temps, l'homme pratique la classification dans sa vie quotidienne, quand il essaie de répondre aux problèmes et questions sur la catégorie des objets dans de nombreuses sciences telles que l'analyse d'images et textes.

De nombreux domaines établissent des classements suivant les objets à catégoriser : les espèces vivantes, les maladies, les produits ou services, les étoiles, les documents d'une bibliothèque, ...

La classification est un outil essentiel pour organiser les connaissances et le travail de chacun au sein de l'ensemble, classer les objets ou les connaissances, c'est dire comment ils/elles se situent les un(e)s par rapport aux autres. Plusieurs points de vue complémentaires peuvent être considérés **[1].**

Ces dernières années ont vu émerger différentes extensions de problème de classification classique, parmi ces derniers se trouvent les problèmes de classification multi-label tel que caractérisation d'une image, d'une musique, ou d'un film.

La classification multi-labels permet d'associer à une observation donnée à une ou plusieurs classes simultanément.

Dans les problèmes de classification multi- label, chaque objet peut appartenir simultanément à plusieurs classes, contrairement aux problèmes standards de classification mono-label dans lesquels un objet appartient à une seule classe. Afin de faire face à de telles tâches d'apprentissage, de nombreuses méthodes ont été proposées pour construire des classifieurs multi-labels. Ces méthodes peuvent être classées en trois

groupes selon la façon dont on traite l'ensemble des données d'apprentissage. Le premier groupe des méthodes transforme le problème d'apprentissage multi-label en un ou plusieurs problèmes d'apprentissage mono-label, le second groupe se base sur l'adaptation directe des algorithmes de classification mono-label pour l'apprentissage multi-label, une méthode fréquemment utilisée pour la classification multi-label et le dernier groupe c'est les méthodes d'ensemble.

2. Notations :

- L'ensemble d'apprentissage d'un problème de classification multi-label est formé de **n** instances étiquetées. Chaque instance d'apprentissage **x** est caractérisée par des attributs, par un ensemble d'étiquettes
- Ensemble fini de classes possibles $Y = \{w_1, w_2, \dots, w_Q\}$
- Ensemble de données d'apprentissage, $D = \{(x_i, Y_i)\ i = 1, \dots, n\}$ où $x_i \in X$ et $Y_i \subseteq Y$
- Ensemble de données test $I = \{(x_j, Y_j)\ j = 1, \dots, m\}$
- Classifieur multi-labels $H: X \longrightarrow 2^Y$ (2^Y est l'ensemble des parties de Y).
- Prédiction $\hat{Y}=H(x)$ ensemble de classes prédites pour l'individu x.

3. Méthodes de classification multi-label :

Les approches de classification multi-label se divisent en trois grandes familles selon **[3]**

1) Méthodes de transformation.
2) Méthodes d'adaptation.
3) Méthodes d'ensembles.

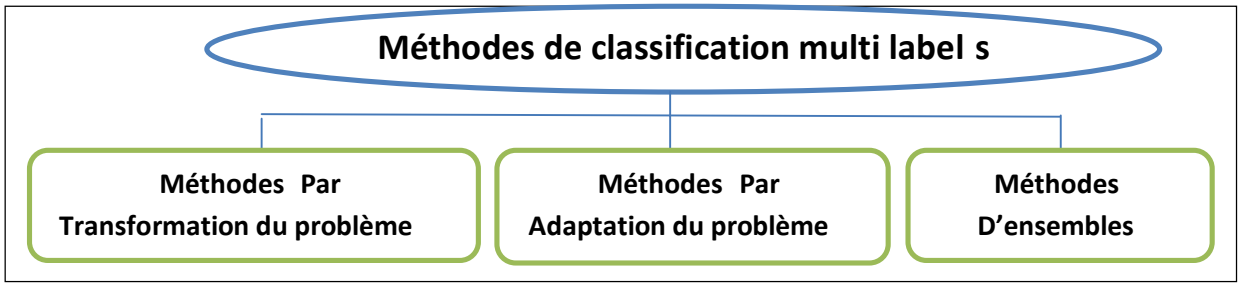


Figure1 : Les Méthodes de classification multi labels

3.1 Méthodes de transformation :

La stratégie la plus simple dans l'apprentissage multi-label est les approches de transformation des problèmes qui peuvent être utilisées avec n'importe quel algorithme d'apprentissage. Dans ces méthodes, le problème de classification multi label est transformé en un ou plusieurs problèmes de classification mono label, qui sont traités séparément en utilisant les approches traditionnelles existantes. Les solutions de ces problèmes sont ensuite combinées pour résoudre la tâche originale de l'apprentissage multi-label. Les méthodes de transformation des problèmes comprennent trois approches principales : la pertinence binaire (BR), le jeu de pouvoir d'étiquette(LP) et le classement d'étiquette(LR).

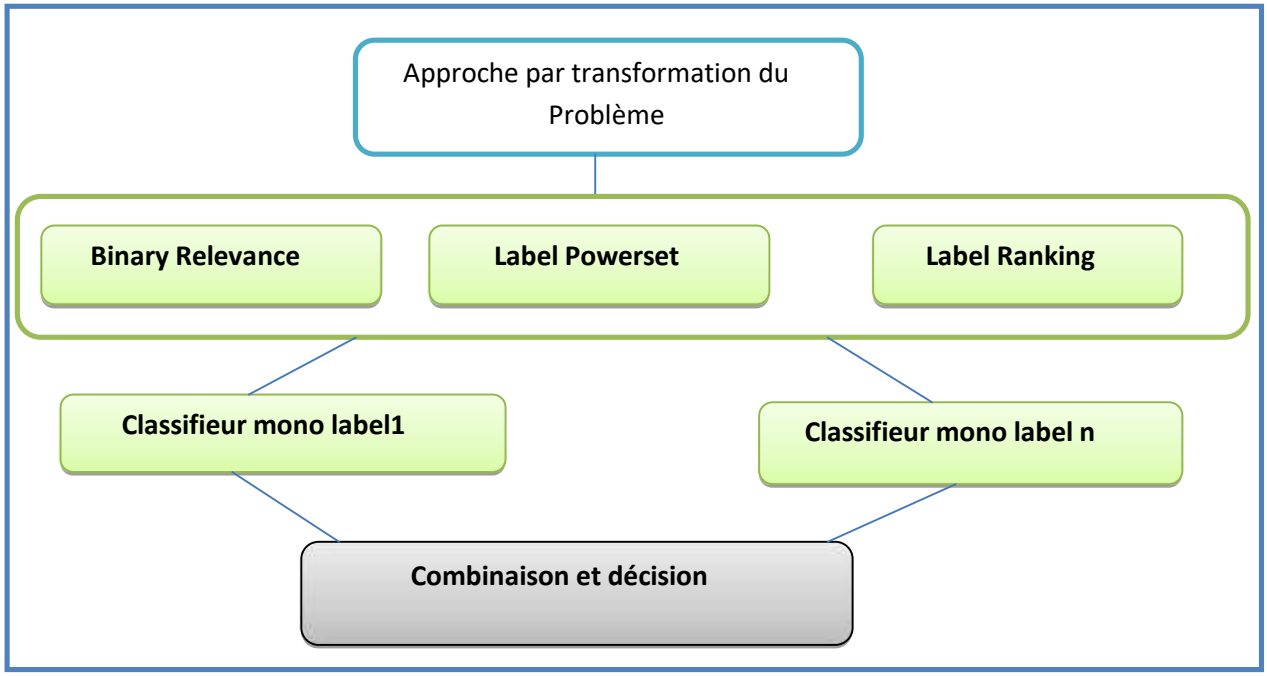


Figure 2 : Méthodes de transformation

3.1.1 La méthode Binary Relevance (pertinence binaire) BR:

- **Définition :**

BR est l'une des approches les plus populaires comme une méthode de transformation qui crée effectivement k ensembles de données (k : le nombre total de classes).

- **Principe :**

L'approche de la pertinence binaire **[4]**, transforme en d classification binaire des problèmes multi labels, un pour chaque variable de classe, $y_1, \ldots y_d$.

Ex	Label
1	$\neg Y_1$
2	Y_1
3	Y 1
4	$\neg Y_1$

(a)

Ex	Label
1	Y 2
2	$\neg Y_2$
3	Y 2
4	Y 2

(b)

Ex	Label
1	Y 3
2	$\neg Y_3$
3	Y 3
4	$\neg Y_3$

(c)

Ex	Label
1	$\neg Y_4$
2	$\neg Y_4$
3	$\neg$ Y 4
4	Y 4

(d)

Figure 3 : Ensembles de données binaires produites par la méthode (BR).

Cette approche associe à chaque classe possible un classifieur binaire, si on a y= [w_1, w_2, w_3], BR créer 3 classifieur mono label pour chaque label existant dans y, par exemple le premier classifieur mono label permet de prédire si w_1 appartient ou non à l'ensemble de classe prédite pour X, ces 3 sorties sont combinées pour fournir l'ensemble des classes prédites.

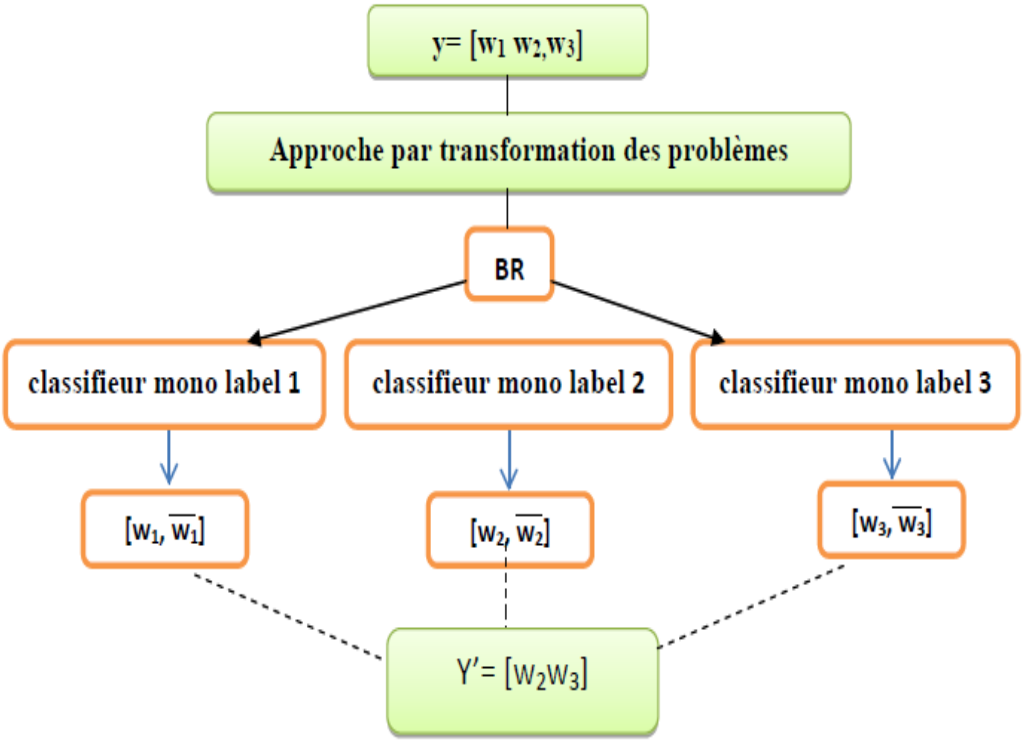
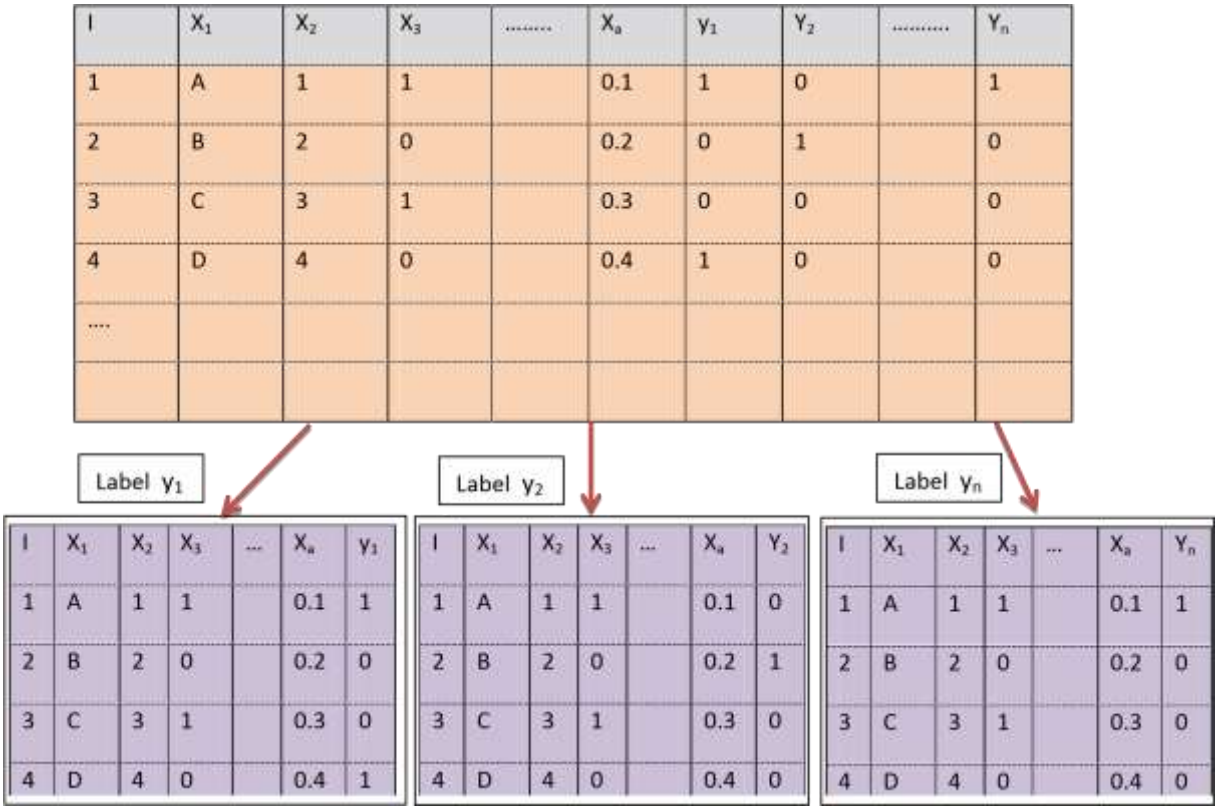


Figure 4 : Méthode de pertinence binaire

Exemple :



I	X_1	X_2	X_3	.....	X_a	y_1	Y_2	.....	Y_n
1	A	1	1		0.1	1	0		1
2	B	2	0		0.2	0	1		0
3	C	3	1		0.3	0	0		0
4	D	4	0		0.4	1	0		0
....									

I	X_1	X_2	X_3	...	X_a	y_1
1	A	1	1		0.1	1
2	B	2	0		0.2	0
3	C	3	1		0.3	0
4	D	4	0		0.4	1

I	X_1	X_2	X_3	...	X_a	Y_2
1	A	1	1		0.1	0
2	B	2	0		0.2	1
3	C	3	1		0.3	0
4	D	4	0		0.4	0

I	X_1	X_2	X_3	...	X_a	Y_n
1	A	1	1		0.1	1
2	B	2	0		0.2	0
3	C	3	1		0.3	0
4	D	4	0		0.4	0

Figure 5 : Exemple de pertinence binaire

La méthode BR a l'avantage d'être simple et peu couteuse en terme de temps de calcul, mais son problème majeur réside dans le fait qu'elle ne tient pas compte des corrélations éventuelles entre les classes.

3.1.2- Label Powerset (LP) :

LP crée un classifieur binaire pour chaque combinaison des labels présente dans l'ensemble d'apprentissage. Par exemple, si les labels possibles pour un exemple étaient A, B et C, la représentation du label powerset de ce problème est un problème de classification à plusieurs classes avec les classes [0 0 0], [1 0 0], [0 1 0], [0 0 1], [1 1 0], [1 0 1], [0 1 1]. [1 1 1] ou par exemple [1 0 1] désigne un exemple où les labels A et C sont présentes et label B est absente.

Exemple : L'approche LP mappe chaque combinaison des labels en une seule label et forme un classifieur mono label.

Ex #	Label set
1	{λ1, λ4}
2	{λ3, λ4}
3	{λ1}
4	{λ2, λ3, λ4}

Ex #	Label
1	1001
2	0011
3	1000
4	0111

Figure 6 : Exemple de la méthode LP

La méthode LP tient compte des corrélations entre les classes, mais elle souffre en général des problèmes de complexité algorithmique.

3.1.3 -Label Ranking (LR) :

Appartient aussi à l'ensemble des méthodes par transformation de problème, elle fait l'apprentissage sur des classifieurs mono-label. Par contre, cette approche permet un classement pour tous les labels appartenant à l'ensemble des labels.

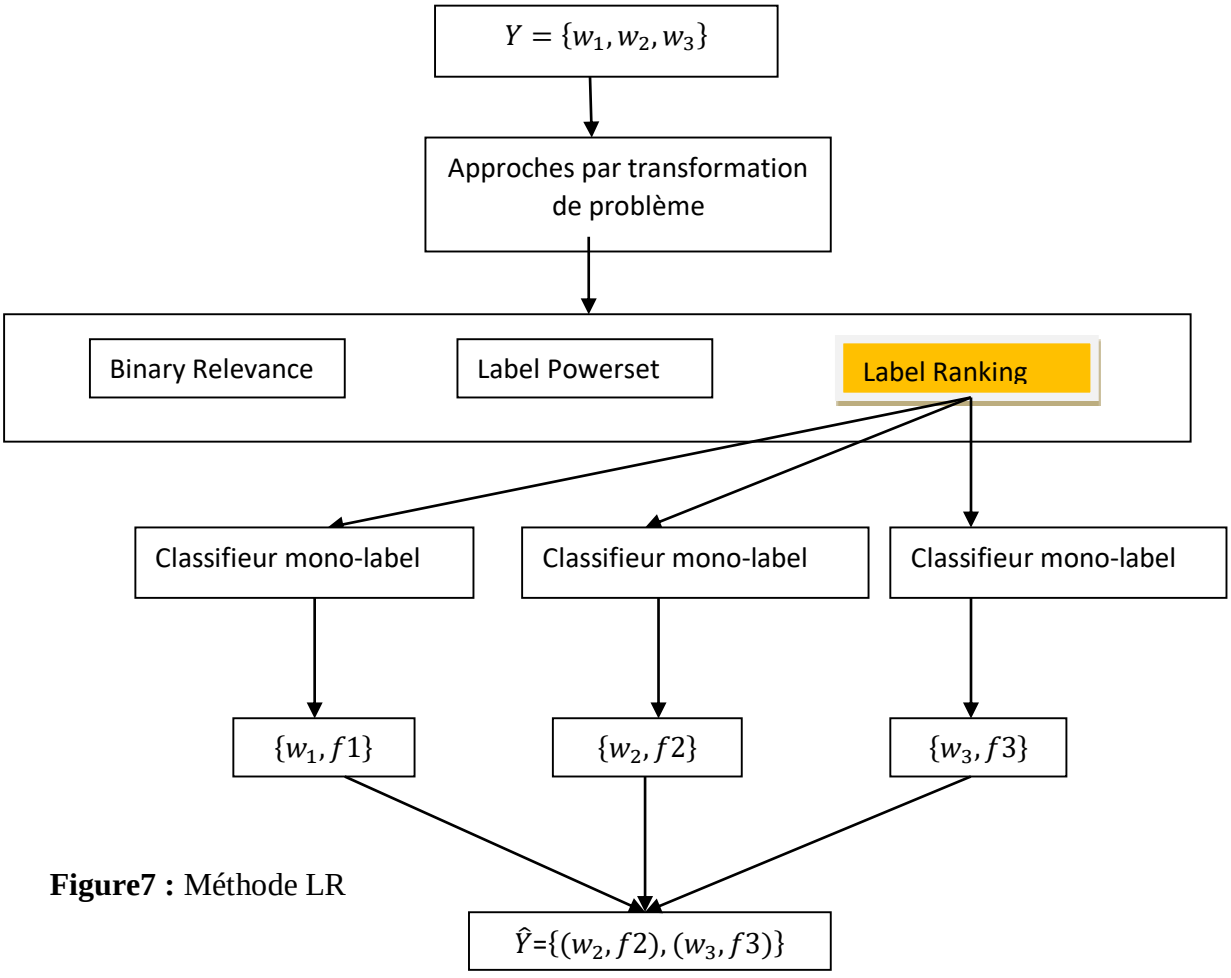


Figure7 : Méthode LR

Exemple :

Etant donné un ensemble de cinq documents possibles (champs d'application) suivants : Mathématique, Physique, Chimie, Biologie et Informatique, $\{M, P, C, B, I\}$, un document peut relever de plusieurs catégories comme Mathématique, Physique et Informatique $\{M, P, I\}$. Du point de vue du classement (Ranking), on peut associer à tout document un ordre sur l'ensemble du champ d'application. Par exemple un document x peut appartenir plus à la classe M que P que I.

Pour résoudre ce problème, différentes méthodes existent. une approche particulière est celle dite préférence par paires qui consiste à apprendre pour chaque couple de labels (w_i, w_j), tel que $i < j$, un classifieur binaire H_{ij} permettant de prédire pour une entrée x si w_i est préférée ou pas à w_j. A noter que n'importe quel classifieur binaire peut être utilisé dans ce cas et que la sortie du classifieur n'est pas nécessairement $\{0,1\}$, elle est généralement comprise dans l'intervalle $[0,1]$. Hûllermier propose d'associer à chaque entrée x une relation floue de préférence R_x définie comme suit :

$$R_x = \begin{cases} H_{ij} & si\ w_i \succ w_j \\ 1 - H_{ij} & sinon \end{cases}$$

Moyennant ces relations de préférences, on calcule pour chaque label w_i, la fonction

$$S_x(w_i) = \sum_{i \neq j} R_x\,(w_i, w_j)$$

Pour une instance x, l'ordre total est obtenu en triant par ordre décroissant les fonctions $S_x(w_i)$ calculées pour chaque élément de Y.

3.1.4 La méthode des chaînes de classifieurs (Classifier chains) CC :

est une méthode alternative pour transformer un problème de classification multi-label en plusieurs problèmes de classification binaire. Elle diffère de la pertinence binaire en ce que les labels sont prédites séquentiellement et la sortie de tous les classifieurs précédents (c'est-à-dire positifs ou négatifs pour un label particulièr) est entrée en tant que caractéristiques des classifieurs suivants. Les chaînes de classifieurs ont été appliquées, par exemple, dans la prédiction de la résistance aux médicaments anti- VIH .

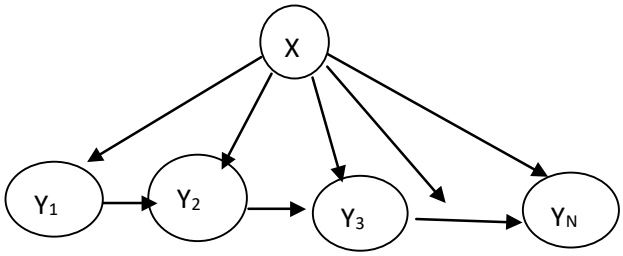


Figure 8 : La représentation graphique de classifier chains (CC)

Essayons de comprendre cela par un exemple. Dans l'ensemble de données ci-dessous, nous avons X comme espace d'entré et Y comme labels.

X	y1	y2	y3	y4
x1	0	1	1	0
x2	1	0	0	0
x3	0	1	0	0

Figure 9 : Ensemble de données

Dans les chaînes de classifieur, ce problème serait transformé en 4 problèmes différents de mono label, comme indiqué ci-dessous. Ici, la couleur jaune est l'espace d'entrée et la partie blanche représente la variable cible.

X	y1
x1	0
x2	1
x3	0

Classifier 1

X	y1	y2
x1	0	1
x2	1	0
x3	0	1

Classifier 2

X	y1	y2	y3
x1	0	1	1
x2	1	0	0
x3	0	1	0

Classifier 3

X	y1	y2	y3	y4
x1	0	1	1	0
x2	1	0	0	0
x3	0	1	0	0

Classifier 4

Figure 10 : Ensemble transformé avec la **c**haîne de classifieurs

1-Tout d'abord, toutes les fonctionnalités (X_1, X_2,, X_m) sont utilisées pour prédire y_1.
2-Ensuite, toutes les fonctionnalités (X_1, X_2,…, X_2, y1) sont utilisées pour prédire y_2.
3-puis, (X_1, X_2,…, X_m, y_1, y_2) sont appliqués pour prédire y_3.
4-Enfin, (X_1, X_2,…, X_m, y_1,y_2,y_3) sont appliqués pour prédire y_4.

L'ordre dans lequel les labels sont prédites à un impact important sur les résultats.

Cette méthode de chaînage transmet les informations des labels entre classifieurs,donc prendre en compte les corrélations dans l'espace d'étiquettes, et surmontant ainsi le problème associé avec BR d'ignorer ces informations.

3.2. Méthodes d'adaptation :

Ces méthodes permettent d'adapter des méthodes de classification mono label existantes dans le domaine d'apprentissage de classification multi label afin de fournir des méthodes de classification multi label **[5].**

Elles personnalisent les algorithmes traditionnels d'apprentissage automatique afin de gérer directement les concepts multi-label. Ces méthodes ont l'avantage de se concentrer sur un algorithme spécifique. Un autre avantage est que ces méthodes utilisent l'ensemble de données d'apprentissage (instances et labels) à la fois pour former un classifieur multi-label. En général, les performances de ces algorithmes sont meilleures dans les problèmes difficiles du monde réel que celles des méthodes de transformation des problèmes, au prix d'une complexité plus élevée. Plusieurs adaptations d'algorithmes d'apprentissage traditionnels ont été proposées dans la littérature, certaines étendant simultanément toutes les labels et instances, comme les arbres de décision et les machines à vecteurs de support multi-label, tandis que d'autres considèrent chaque classe séparément, comme la méthode multi label des voisins les plus proches (MLkNN).

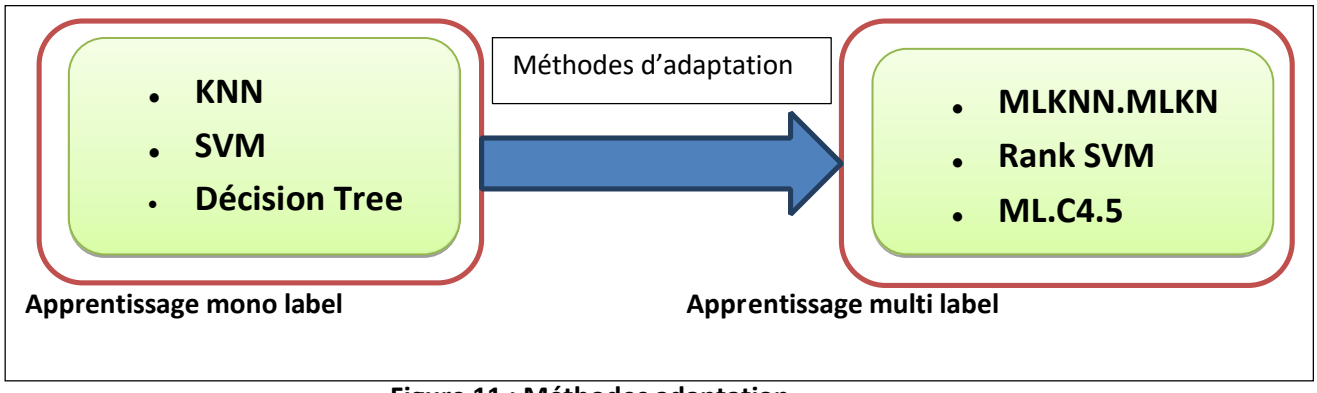


Figure 11 : Méthodes adaptation

3.2.1. MLkNN :

Pour chaque instance de l'ensemble de test, ses K voisins les plus proches de l'ensemble de trains sont identifiés. Pour chaque classe y dans Y, le nombre d'instances voisines appartenant à y est utilisé pour calculer les probabilités a posteriori auxquelles l'instance de test appartient ou n'appartient pas

à y. En fonction des probabilités les plus grandes, nous décidons d'affecter ou non la classe y à une instance de test.

Pour une meilleure compréhension, supposons que les échantillons ci-dessous sont des voisins trouvés dans l'ensemble d'apprentissage pour l'instance de test X = 1.

X	Y_1	Y_2
1	f_1	f_2
2	f_1	f_3
3	f_2	f_4
1	?	?

Figure 12 : échantillons des voisins de X

Nous devons maintenant calculer les probabilités a priori et a posteriori pour chaque classe pour l'instance de test X = 1

- $P(f_1 \mid x=1) = P(f_1).\ P(x = 1 \mid f_1) = 2/3.\ ½ = 1/3$
- $P(f_2 \mid x=1) = P(f_2).\ P(x = 1 \mid f_2) = 2/3.\ ½ = 1/3$
- $P(f_3 \mid x=1) = P(f_3).\ P(x = 1 \mid f_3) = 1/3.\ 0 = 0$
- $P(f_4 \mid x=1) = P(f_4).\ P(x = 1 \mid f_4) = 1/3.\ 0 = 0$

Des valeurs maximales ont été obtenues pour f_1 et f_2. Par conséquent, nous attribuons une instance de test à ces deux classes.

- **Avantages et inconvénients :**
  - Meilleure précision que les méthodes précédentes.
  - Les corrélations entre les étiquettes sont considérées.
  - Les distributions de données pour certaines étiquettes sont déséquilibrées.

3.3. Méthodes d'ensemble (hybride) :

Pour faire des prédictions multi-labels ; leurs classifieurs de base appartiennent à des méthodes de transformation de problèmes ou d'adaptation d'algorithmes.

Dans l'apprentissage multi-label, les exemples étiquetés sont généralement annotés par des experts. Dans de nombreuses applications, il n'existe aucune vérité fondamentale pour

attribuer sans ambiguïté un ensemble de labels à une instance. Un expert peut ne pas être sûr de l'étiquetage d'une instance. De plus, plusieurs experts peuvent avoir des opinions contradictoires, ce qui peut prêter à une confusion lors de l'apprentissage à partir de l'ensemble de données obtenu. Par exemple, dans l'exemple de prédiction des émotions, un expert peut décider qu'une chanson évoque le bonheur et la relaxation, tandis qu'un autre peut juger que cette musique peut induire la nostalgie. Ces problèmes pourraient introduire une certaine incertitude dans le processus d'étiquetage.

Les méthodes d'ensemble sont basées sur les transformations et l'adaptation du problème afin de fournir un classifieur multi label.

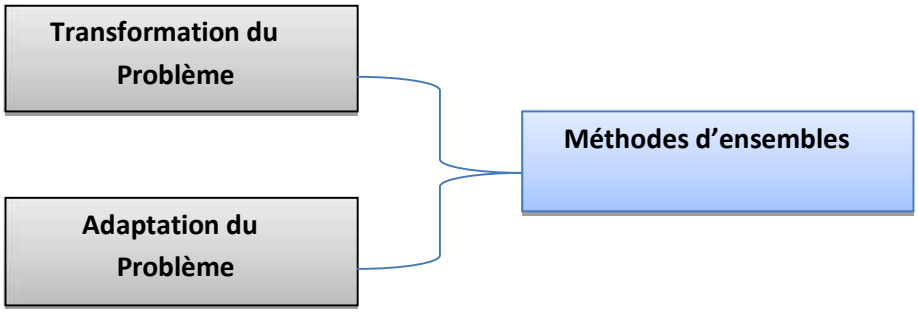


Figure 13 : les bases de la méthode d'ensemble.

Exemple :

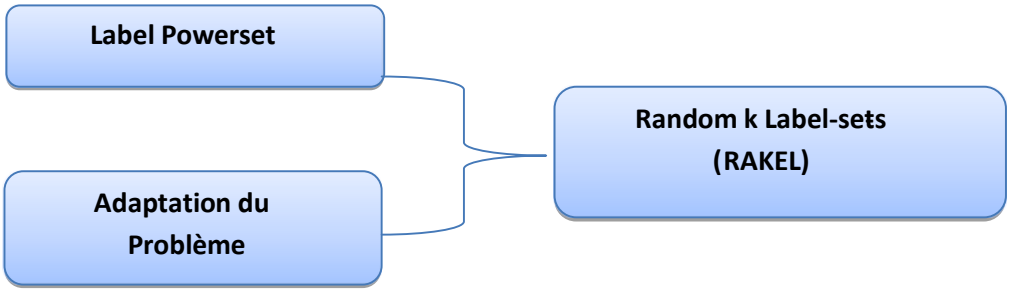


Figure 14: Exemple d'une méthode d'ensemble.

3.3.1. Ensemble de chaînes classifier (ECC) :

Ensemble of Classifier Chains (ECC) utilise Classifier Chains (CC) comme classifieurs de base. ECC a été proposé pour soulager l'effet de l'ordre des classifieurs dans CC, en formant un ensemble de classifieurs CC, C1, . . ., Cm. Chaque Ck peut être formé avec un ordre de chaîne aléatoire sur un sous-ensemble aléatoire de D.

Étant donné une instance non classée, les prédictions de différents classifieurs sont rassemblées et combinées pour chaque label afin que chaque label reçois un nombre de votes. Pour sortir le jeu multi label final, un seuil est utilisé pour sélectionner les labels les plus pertinentes. Cette méthode d'ensemble peut utiliser n'importe quelle méthode de transformation de problème multi-label comme classifieur de base. En général, ECC surpasse BR et CC tout en conservant une complexité de calcul acceptable.

3.3.2.RAKEL:

Une autre variation est le *k* aléatoire-labelsets (RAKEL) algorithme, qui utilise plusieurs classifieurs LP, chacun formé sur un sous-ensemble aléatoire des labels réelles; la

prédiction des labels est ensuite réalisée par un schéma de vote. Un ensemble de classifieurs multi-labels peut être utilisé de la même manière pour créer un classifieur d'ensemble multi-label. Dans ce cas, chaque classifieur vote une fois pour chaque label qu'il prédit plutôt que pour un seul label.

4. Les étapes d'une classification :

1. Choix des données.
2. Calcul des similarités entre les n individus à partir des données initiales.
3. Choix d'un algorithme de classification et exécution.
4. L'interprétation des résultats :

    - évaluation de la qualité de la classification
    - description des classes obtenues.

5.Les mesures de performances :

L'évaluation des performances des systèmes d'apprentissage à multi label diffère de celle de la classification à mono label.

Un bon classifieur est caractérisé par :

- Une bonne performance : taux élevé de bonne classification
- Possibilité de généralisation : bonne performance sur des données différentes de celles utilisées en apprentissage.
- Rapidité : vitesse de classification, apprentissage
- Robustesse : insensibilité au bruit.
- Classification avec des données manquantes.
- Incrémentation.

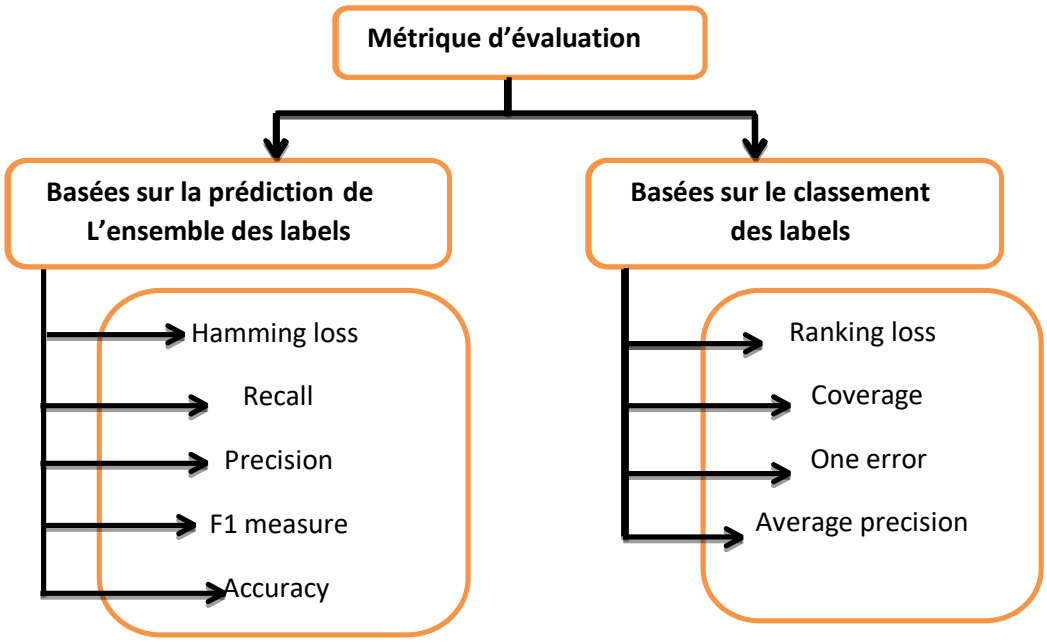


Figure 15: les mesures de performances

5.1 Mesures basées sur la prédiction de L'ensemble des labels :

Soit l'ensemble test T= {(x_i, y_i), i_1, …, N}, avec Yi le vrai ensemble label associé à l'ensemble test x_i et soit Y'_i l'ensemble des labels prédits par le classifieur multi label pour x_1, soit H un classifieur multi-label et N le nombre des exemples multi-label.

5.1.1 Accuracy :

Cette métrique mesure le degré de similarité entre Yi et $|\hat{Y}i|$. Il permet d'évaluer, pour chaque exemple test $x_i \epsilon$ T, la similarité de Jaccard entre le vecteur des vrais labels Yi et le vecteur des labels prédits $|\hat{Y}i|$, plus grande est la valeur de la précision, plus grande est la performance.

Il est défini par :

$$\textbf{Accuracy (H ,N)} = \frac{1}{N}\sum_{i=1}^{N}\frac{|Yi \cap \hat{Y}i|}{|Yi \cup \hat{Y}i|}$$

5.1.2 Recall :

Il permet d'évaluer, pour chaque exemple test $x_i \in$ T, la proportion des labels correctement prédits sur l'ensemble des vrais labels y_i, il est défini par :

$$\textbf{Recall(H,N)} = \frac{1}{|N|}\sum_{i=1}^{|N|}\frac{|Yi \cap \hat{Y}i|}{|Yi|}$$

5.1.3 Precision:

Il permet d'évaluer, pour chaque exemple test $x_i \in T$, la proportion des labels

Correctement prédits sur l'ensemble des labels prédits positifs, i.e. le taux de bonnes prédictions, il est défini par :

$$\textbf{Precision(H,N)} = \frac{1}{|N|}\sum_{i=1}^{|N|}\frac{|Yi \cap \hat{Y}i|}{||\hat{Y}i||}$$

5.1.4 F. mesure « F1 » :

La F-mesure correspond à une moyenne harmonique de la précision et du rappel.

$$\textbf{F1(H,N)}= \frac{2}{\frac{1}{Precision(H,N)}+\frac{1}{Recall(H,N)}}$$

5.1.5 Le coût de Hamming :

Ce critère évalue combien des labels sont mal classées. Plus la valeur de hamming_loss est faible, meilleure est la performance. La performance est parfaite lorsque hamming_loss(h)=0.

$$\textbf{C.H}= \frac{1}{N}\sum_{i=1}^{N}\frac{|Yi \Delta \hat{Y}i|}{l}$$

Où: Δ est la différence symétrique entre deux ensembles

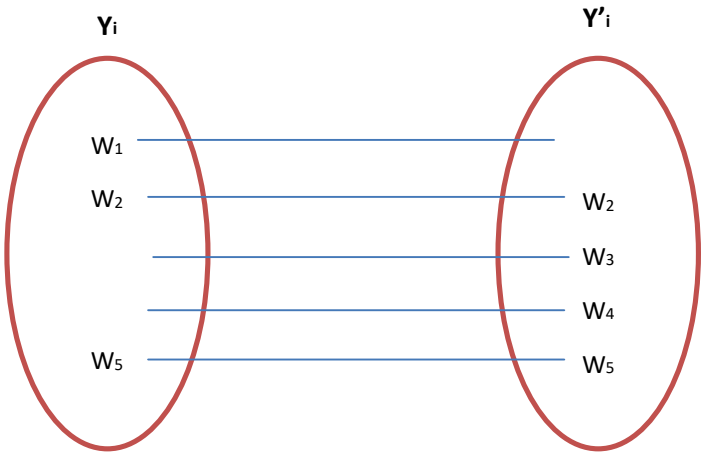
l : le nombre total des classes.

$|Yi\Delta\hat{Y}i|$: le nombre des classes prédites mal classés (nombre des erreurs).

N : nombre des exemples.

Exemple :

Pour un individu x les vraies classes sont w_1, w_2, w_5 alors que l'ensemble des classes prédites sont w_2, w_3, w_5, w_4 ce classifieur contient 3 erreurs.



Figue16: exemple du hammingloss

Pour un exemple donc N=1

L=5 (le nombre total des classes).

$|Yi \Delta \hat{Y} i|$=3 (nombre des erreurs).

Donc : HLoss = 3 /5

5.2 Mesures basées sur le classement des labels :

5.2.1 Le coût de classement (RankingLoss) :

Ce critère mesure combien de fois un label incorrect est mieux classé par rapport à un label correct par exemple si l'ensemble des vraies classes contient w_1, w_2, w_5 et l'ensemble des classes prédites contient w_3, w_4.

Cette mesure à un sens avec les classifieurs qui utilise un rang de fonction de score qui a un intervalle de [0…1].

Exemple :

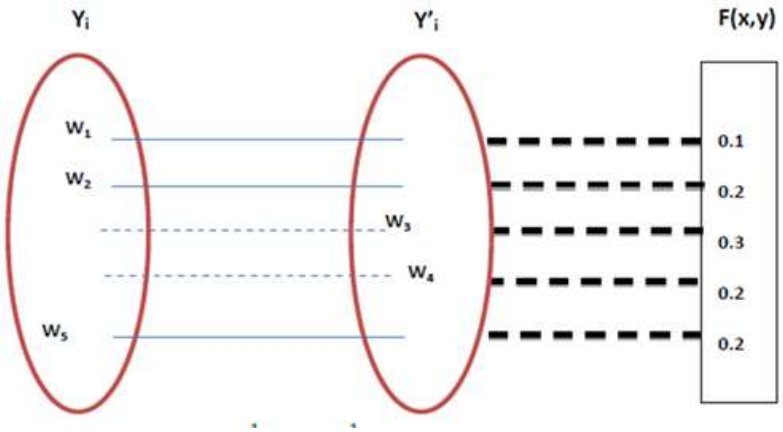


Figure 17 : exemple de RankingLoss

$$\textbf{RLoss(f,x)} = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{|Yi|\,|\hat{Y}i|} * nombre\ des\ erreurs$$

Pour un exemple **n=1** :$|Y_i| = 3$ $|\hat{Y}_i| = 2$

On compare les rangs des vraies classes (1) avec les rangs des classes prédites (2),

si

(1) ≤ (2) donc err

(2) eur

- F (x, w_1) ≤ F (x, w_3) ⇒ erreur.
- F (x, w_1) ≤F (x, w_4) ⇒ erreur.
- F (x, w_2) ≤ F (x, w_3) ⇒ erreur.

- F (x, w2) ≤ F (x, w_4) ⇒ erreur.
- F (x, w_5) ≤F (x, w_3) ⇒ erreur.
- F (x, w_5) ≤ F (x, w_4) ⇒ erreur.

Donc :

RLoss(f,x) = $\frac{1}{1}\sum_{i=1}^{n}\frac{1}{3*2} * 6 = 1$

5.2.2 One-error :

Évalue combien de fois l'étiquette de premier rang n'est pas dans l'ensemble des labels pertinents de l'exemple. La métrique one_error(f) prend des valeurs entre 0 et 1. Plus la valeur de one_error(f) est faible, meilleure est la performance. Cette mesure d'évaluation est définie comme :

One_error(f) = $\frac{1}{N}\sum_{i=1}^{N}[\![[arg\max_{y\,\epsilon Y} f(xi, Y)] \notin Yi]\!]$

où y ϵ L= {y_1, y_2, …, y_Q} et [[ω]] est égale à 1 si ω se maintient et 0 autrement pour tout prédicat ω. Notez que, pour les problèmes de classification mono-label, one_error est identique à une erreur de classification ordinaire.

5.2.3 Coverage « la couverture » :

Évalue jusqu'à quel point, en moyenne, nous devons descendre dans la liste des labels classés afin de couvrir tous les labels pertinents de l'exemple. Plus la valeur de coverage(f) est faible, meilleure est la performance :

$$\mathbf{coverage(f)} = \frac{1}{N}\sum_{i=1}^{N}\max_{y\epsilon Yi} rankf\,(xi, y) - 1$$

où rankf(x_i, y) mappe (couvre) les sorties de f(**x**i, y) pour tout y ϵ L = {y1, y2, …, yQ} de sorte que f(x_i, y_m) > f(x_i, y_n) implique rankf(x_i, y_m) < rankf(x_i, y_n). La plus petite valeur possible pour la coverage(f) est lc, c'est-à-dire la cardinalité de l'étiquette au niveau de la base de données considérée.

6. Quelques applications de la classification multi label :

Ces toutes dernières années, plusieurs ateliers ont été organisés et des numéros spéciaux ont été édités sur le thème de l'apprentissage multi-label. La question de l'apprentissage à partir de données multi-label a récemment attiré l'attention de nombreux chercheurs, motivés par un nombre croissant de nouvelles applications. Parmi les principales applications de l'apprentissage multi-label

- **La catégorisation de texte**:

Elle consiste à attribuer un ensemble de catégories prédéfinies aux documents. Comme un document peut appartenir simultanément à plus d'une catégorie, il peut être abordé en apprentissage multi-label.

- **Multimédia :**

Les techniques de l'apprentissage multi-label ont été appliquées à de nombreux types de ressources telles que les images, vidéos et sons. Parmis les exemples d'applications : annotation automatique des images **[6]**, Face **[7]**, annotation vidéo **[8]**, détection des émotions dans la musique **[9]**, extraction des métadonnées musicales **[10]** et classification des émotions de la parole **[11]**.

Exemple : Quand nous entendons un morceau de musique classique sur l'autoradio, comment on se sent ? Nous sentons-nous heureux, tristes, étonnés, calmes ou en colère ?

- **La biologie :**

Il convient de noter la prédiction des fonctions génétiques **[12]** et la prédiction des fonctions protéiques **[13]**ainsi que d'autres applications telle la prédiction des structures 3D des protéines **[14].**

- **Analyse des données chimiques**:

L'apprentissage multi-label a également été appliqué pour prédire les effets indésirables des médicaments et pour identifier les médicaments qui ont deux ou plusieurs actions biologiques différentes (découverte de médicaments) **[15],**et pour détecter les contaminants dans les lubrifiants de machines en utilisant l'analyse des images spectrales **[16].**

- **L'exploitation des réseaux sociaux**:

C'est devenu un nouveau domaine d'intérêt. L'apprentissage comportemental collectif consiste à inférer le comportement ou les préférences des individus **[17].** La publicité en réseau social ou l'annotation automatique des nœuds d'un graphe multi-relationnel partiellement marqué sont d'autres champs d'application **[18].**

- **E-Learning** :

L'apprentissage multi-label a également été appliqué pour classer les styles d'apprentissage sur la base des profils des apprenants et pour marquer les objets d'apprentissage **[19].**

- **Autres applications** :

D'autres domaines d'application intéressants sont le marketing direct, où les acheteurs potentiels de certains produits sont identifiés et le diagnostic médical (de nombreux symptômes peuvent être associés à plus d'un syndrome) **[20].** Dans la référence **[21],** l'apprentissage multi-label a été appliqué pour classer les images de dermoscopie de lésions cutanées qui pourraient contenir plusieurs lésions de modèle.

Conclusion :

Dans ce chapitre, nous avons présenté les méthodes pour l'apprentissage multi-label. Le sujet de l'apprentissage multi-label a récemment reçu un effort de recherche significatif. Il a également attiré beaucoup d'attention de la part des chercheurs.

Les méthodes multi-label qui ont été proposées dans la littérature, sont divisées en trois groupes principaux : la transformation des problèmes, l'adaptation d'algorithmes et les méthodes d'ensemble

La variété des mesures d'évaluation est nécessaire pour donner une vue d'ensemble de la performance des algorithmes sous différents angles. Les différentes mesures d'évaluation qui sont généralement utilisées dans le contexte de l'apprentissage multi-label, sont réparties en deux groupes : cinq mesures basées sur la prédiction de l'ensemble des labels, quatre mesures sur le classement. Les mesures basées sur la prédiction de l'ensemble des labels sont le plus largement utilisées pour la classification multi-label. Pour évaluer la capacité des classifieurs choisis, nous avons sélectionné 5 mesures basées sur la prédiction de l'ensemble des labels (Hamming-loss, accuracy, precision, recall et F-mesure).

Ce chapitre avait pour but de synthétiser les méthodes d'apprentissage multi-label avec un aperçu sur leur prédiction, actuellement disponibles dans la littérature, ceci en contexte supervisé. Dans le chapitre suivant, on s'intéressera à la classification multi label hiérarchique ou les classes sont organisées sous forme d'une hiérarchie.

Chapitre 2

La classification multi label hiérarchique

Introduction :

La classification est une tâche d'apprentissage automatique qui vise à construire des modèles de distribution de classe en tenant compte d'un ensemble d'attributs prédictifs (également appelés fonctionnalités). Le résultat d'un tel modèle est utilisé pour attribuer des labels à de nouveaux exemples dont les seules informations connues sont les valeurs des attributs prédictifs.

La plupart des problèmes de classification traités considèrent que chaque exemple n'est associé qu'à une seule étiquette de classe. De plus, les étiquettes de classe sont traitées comme indépendantes, dans le sens où les algorithmes de classification ne supposent pas l'existence d'une quelconque relation entre les différentes étiquettes de classe.

Néanmoins, dans de nombreux problèmes de classification du monde réel par exemple, dans la prédiction de la fonction de gène, la tâche consiste à classifier ses fonction (classes)en se référant à une hiérarchie de fonctions prédéfinies, comme Gene Ontologie (GO). Une instance qui appartient à une classe appartient automatiquement à toutes ses superclasses. **La Figure 18** illustre le procédé de prédiction de la fonction de gènes.

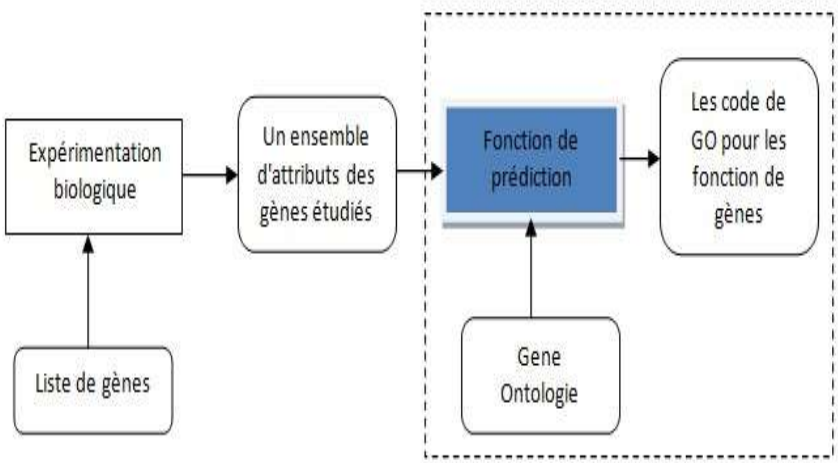


Figure 18 : Prédiction des fonctions de gènes en utilisant Gene Ontology (GO)

Dans la prédiction de la fonction de gènes, le nombre de classes fonctionnelles est très important, et un gène peut appartenir à plusieurs classes simultanément, Dans ce cas, les classes forment une structure hiérarchique **[22].**

1. Classification multi labels hiérarchique :

La classification multi label hiérarchique est une tâche de classification complexe dans laquelle les classes impliquées dans le problème sont structurées hiérarchiquement et chaque exemple peut appartenir simultanément à plus d'une classe dans chaque niveau hiérarchique. Ce domaine de recherche n'a pas eu une grande attention, et pourtant on retrouve ce problème dans beaucoup d'applications importantes telle que les bibliothèques numériques, Le domaine de la bio-informatique et La reconnaissance d'images.

La classification hiérarchique devient une nécessité dans de nombreux domaines **[23][24][25][26][27][28][29][30]**. En effet, lorsque le nombre de classes est important (très grand), il est souvent utile de les organiser en groupes et sous-groupes. Par exemple, de grandes collections de pages web peuvent être organisées en hiérarchie comme le montre la Figure 19. Cette organisation hiérarchique est essentielle parce qu'elle permet à l'utilisateur de se concentrer sur le niveau approprié de détails.

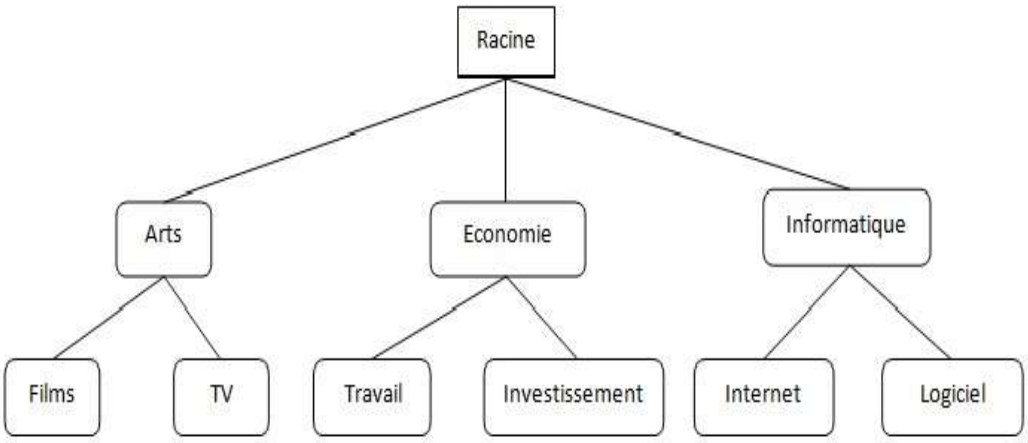


Figure 19 : La partie supérieure du projet Open web directory dont la hiérarchie est organisée en arbre.

Seuls quelques travaux ont traité ce sujet, la plupart d'entre eux présentent certaines limites. Par exemple, certains travaux **[31, 32, 33, 34]** ignorent la relation hiérarchique entre les classes et ils transforment le problème de classification hiérarchique en un problème à un seul niveau. D'autres travaux **[35, 36]** s'appuient sur une structure hiérarchique arborescente (sous forme d'un arbre) où chaque classe a au maximum un seul parent (Figure 21). Cependant, dans de nombreuses applications du monde réel, les classes peuvent avoir plus d'un parent, dans ce cas les relations inter-classes ne peuvent être représentées que par un graphe dirigé acyclique (DAG), comme représenté dans la Figure 20.

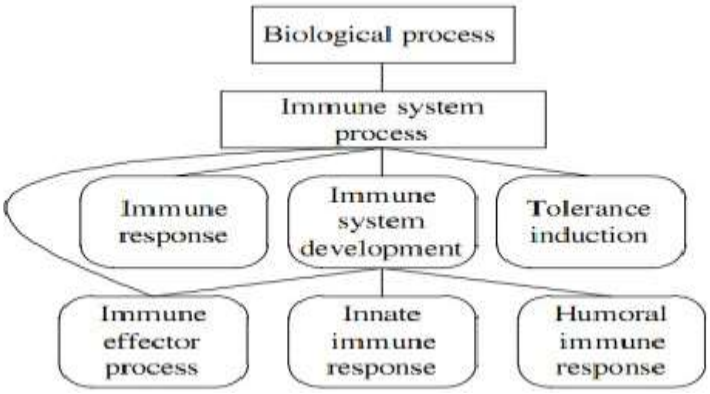


Figure 20 : Partie supérieure du système immunitaire dans Gene Ontology (GO) dont la structure est un graphe dirigé acyclique

1.1 Énonce du problème :

Compte tenu d'un espace d'exemples $\mathcal{X}$, l'objectif du processus de formation est de trouver une fonction capable de mapper chaque exemple $\mathcal{X}$ en un ensemble de classes, en respectant les contraintes de la structure hiérarchique et en optimisant un critère de qualité.

Bien que la définition donnée de la console HMC indique qu'un modèle induit doit classer un exemple dans des chemins hiérarchiques appropriés, certains travaux permettent des prédictions incohérentes. C'est le cas dans les études réalisées par Obozinski et Valentini **[37],** où des prédictions incompatibles avec la hiérarchie sont faites, suivies d'une étape supplémentaire de mise en cohérence des affectations de classe avec la hiérarchie.

La Figure 21, donne un cas d'affectation de classes cohérent et un autre incohérent. La Figure 21.a donne un cas d'affectation cohérent, l'instance $\mathcal{X}_i$ qui appartient aux classes $\mathcal{C}_4$ et $\mathcal{C}_5$ est étiquetée avec toutes les classes ancêtres, $\mathcal{C}_1$et $\mathcal{C}_2$ Tandis que la Figure 21.b montre un cas d'affectation de classes incohérent, l'instance xi appartient à la classe $\mathcal{C}_5$, mais elle n'est pas affectée à la classe $\mathcal{C}_2$ qui est le parent de la classe $\mathcal{C}_5$. De telles situation peuvent se produire si on applique les méthodes de classification classiques aux tâches hiérarchiques, sans modifications.

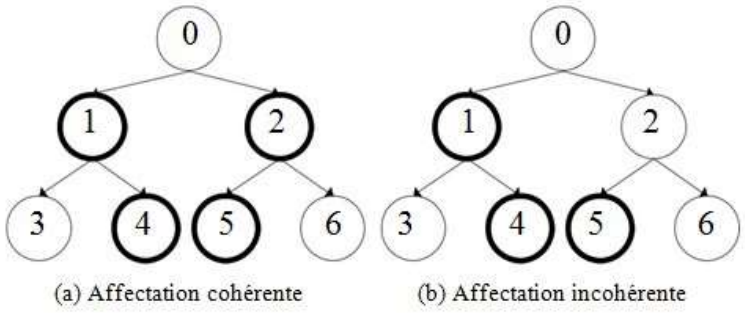


Figure 21 : Exemple d'affection cohérente et incohérente, les cercles en gras représentes les classes affectées par le classifieur à une instance x_i.

1.2 Particularités de la classification hiérarchique :

Hiérarchie ($\mathcal{H}$) :

Une hiérarchie H = ($\mathcal{N}$, ξ) est défini comme un graphe dirigé acyclique (DAG) constitué d'un ensemble de nœuds $\mathcal{N}$ et d'un ensemble d'arêtes ξ, où une arête est une paire de noeuds ordonnés, (N_p, N_c) est défini à partir du nœud parent N_p au nœud enfant N_c, spécifié par l'opérateur $N_p \Rightarrow N_c$ qui est également appelé le chemin direct de N_p à N_c.

Un chemin $N_a \rightarrow N_c$ indique qu'il existe un chemin à partir du nœud ancêtre N_a vers le nœud enfant N_c.

A noter que dans une hiérarchie H avec un chemin $N_a \rightarrow N_c$ le chemin inverse $N_c \rightarrow N_a$ n'existe pas, puisque la hiérarchie est acyclique.

En outre, il existe un nœud appelé nœuds racine N_r d'un graphe H qui n'a pas de parents, les nœuds qui n'ont pas de nœuds enfants sont appelés nœuds feuilles, tous les nœuds à part le

nœud racine et les nœuds feuilles sont appelés nœuds internes. Dans la Figure 22, le nœud racine est {0}, les nœuds internes sont {1,2} et les nœuds feuilles sont {3,4,5,6} :

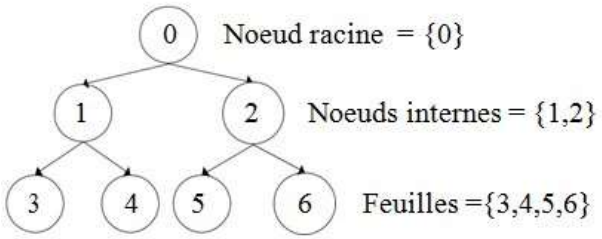


Figure 22: Hiérarchie sous forme d'une structure arborescente

La structure des hiérarchies peut être classée en deux catégories principale, arborescente « Figure 22 » et graphe dirigé acyclique (DAG) « Figure 23 ». La principale différence entre la structure arborescente (en arbre) et la structure DAG est que chaque nœud dans la hiérarchie arborescente a juste un seul parent, tandis que dans la hiérarchie DAG chaque nœud peut hériter de plusieurs parents, comme le montre la figure suivante :

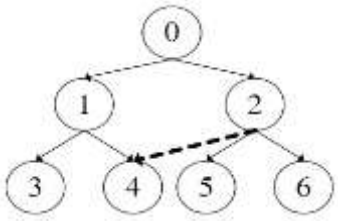


Figure 23 : Hiérarchie sous forme d'une structure DAG

Classes ($\mathcal{C}$) :

Soit $\mathcal{C}$ un ensemble fini de classes (labels) et chaque classe $\mathcal{C}_i$ est constituée d'un ensemble d'instances (exemples) $\mathcal{X}$. La relation entre les classes est définie dans la hiérarchie $\boldsymbol{\mathcal{H}}$ où chaque nœud N_i correspond à une et une seule classe C_i, ($\mathcal{C} \equiv \mathcal{N} \in \boldsymbol{\mathcal{H}}$).

Instances ($\mathcal{X}$) :

Soit R^p un espace d'instances où chaque instance est décrite par un vecteur de p attributs. Soit $\mathcal{X} \subset R^p$ un ensemble fini d'instances. Dans la hiérarchie $\boldsymbol{\mathcal{H}}$ un ensemble d'instances est affecté à une ou plusieurs classes dans $\mathcal{C}$.

Relations entre classes :

$\mathcal{H}$ est une structure hiérarchique définissant les relations entre toutes les classes dans $\mathcal{C}$. L'hypothèse est que $C_{parent} \longrightarrow C_{Enfant}$ est défini comme étant une relation "EST UN" (IS A) dans la structure arborescente (structure en arbre) et comme une relation "PARTIE DE" dans la structure DAG.

Il est important de définir ce qu'est exactement une taxonomie de classes. Un arbre structuré est un ensemble partiellement ordonné $(\mathcal{C}, \prec)$ où $\mathcal{C}$ est un ensemble fini de classes et la relation $\prec$ représente la relation "EST UN". Cette relation, doit respecter les propriétés suivantes **[36]** :

Le plus grand élément "*R*" est la racine de l'arbre.

- **Asymétrique :**

  $\forall C_i, Cj \in$ C, si $C_i \prec C_j$ $(C_i \longrightarrow C_j)$ alors $C_j \not\prec C_i$. Par exemple, tous les lions sont des mammifères, mais les mammifères ne sont pas tous des lions.

- **Anti-reflexive** *:*

  $\forall C_i \in$ C, $C_i \not\prec C_i$. Par exemple, le parent de la classe lion ne peut pas être elle-même.

- ***Transitive :***

  $\forall C_i, C_j, C_k \in$ C, $C_i \prec C_j$ et $C_j \prec C_k$ implique $C_i \prec C_k$. Par exemple, tous les lions sont des mammifères et les mammifères sont des animaux, par conséquent, tous les lions sont des animaux.

Cette définition, bien que proposée, à l'origine pour les structures arborescentes, peut être aussi utilisée pour les classes structurées en DAG.

2. Méthodes de classification multi labels hiérarchique :

Selon **[38]** et**[39],** La classification hiérarchique diffère selon un certain nombre de critères :

- **Le premier critère :** concerne le type de la structure hiérarchique utilisée. Soit une structure arborescente ou une structure en DAG.
- **Le deuxième critère :** est lié à la façon dans la classification est effectuée au sein de la structure hiérarchique :

  - La méthode de classification hiérarchique peut être mise en œuvre de façon à ce que toute instance est affectée impérativement aux nœud (classes) feuilles(NMLNP) **[39].**

- La méthode de classification hiérarchique peut envisager d'arrêter le processus de classification à n'importe quel nœud de n'importe quel niveau de la hiérarchie.

➢ **Le troisième critère :** est lié à la façon dont la structure hiérarchique est exploitée. Dans **[40]** les auteurs ont résumé les trois approches existantes dans le domaine de la classification hiérarchique, incluant, l'approche de classification plate, l'approche de classification locale (Top-down) et l'approche globale (Big bang)..

2.1 Approche de classification plate (Flat classification):

L'approche la plus simple est de transformer le problème hiérarchique en un problème de classification plate **[31, 32, 33, 34]** .Cette approche ne tient pas compte de la hiérarchie de classes, et ne traite qu'avec les noeuds feuilles, soit en utilisant un seul classifieur multi-label ou un ensemble de classifieurs binaires (un pour chaque nœud feuille).La figures 24 donne un scénario de cette approche, seules les classes {1.1,1.2,2.1.1,2.1.2,2.2.1,2.2.2}sont considérées.

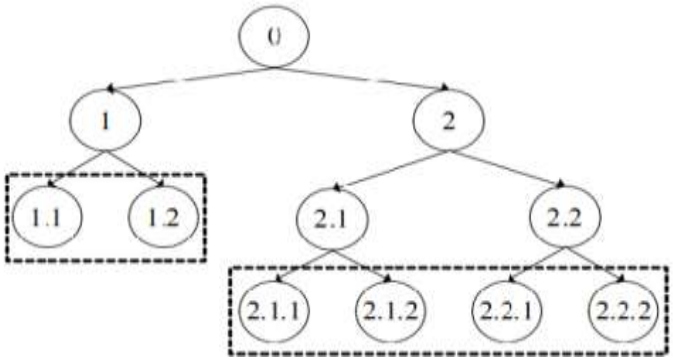


Figure 24: Approche de classification hiérarchique plate

2.2 Approche de classification locale (Top-down):

C'est l'approche la plus utilisée dans le domaine de la classification hiérarchique **[40]**. Un ou plusieurs classifieurs sont formés pour chaque nœud (classe) ou pour chaque niveau de la hiérarchie. On obtient ainsi un arbre de classifieurs dans la structure en arbre ou un graphe de classifieurs dans la structure DAG. Ces classifieurs sont construits à partir d'informations locales.

Elle est aussi appelée "approche Top-down"**[36]** parce que les phases d'apprentissage et de test se déroulent de façon descendante.

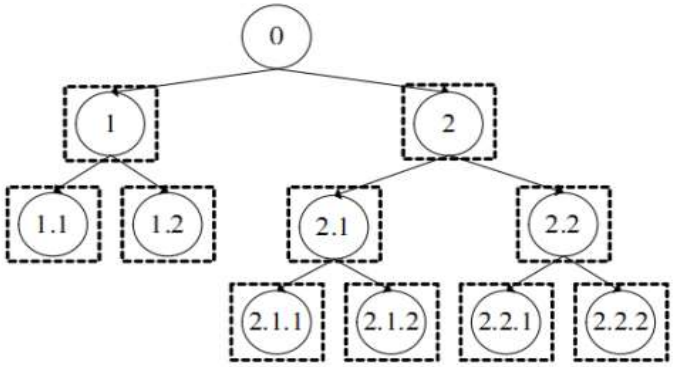


Figure 25: Approche de classification locale

2.2.1. Phase de préparation des données d'apprentissages :

Dans **[41]** et **[42]**, Les auteurs proposent des études comparatives des différentes méthodes d'attribution d'un ensemble de données d'apprentissage . Ces méthodes sont classées en cinq catégories :

L'exemple suivant sera utilisé pour montrer comment construire un ensemble d'apprentissage pour la classe $C_{2.1}$ de la hiérarchie représentée par la Figure 26.

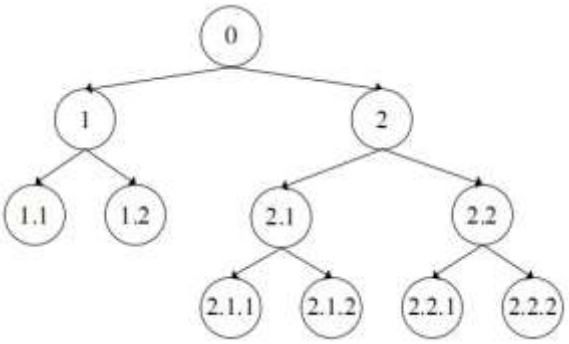


Figure 26 : Exemple d'une hiérarchie de classes

Tableau 1 : Les notations sont définies dans le tableau suivant :

Symboles	*Signification*
Tr	L'ensemble de toutes les données d'apprentissage
$Tr^{+}(C_i)$	L'ensemble des instances positives pour C_i dans l'ensemble d'apprentissage
$Tr^{-}(C_i)$	L'ensemble des instances négatives pour C_i dans l'ensemble d'apprentissage
$\uparrow (C_i)$	Les classes mères (parents) de la classe C_i
$\downarrow (C_i)$	Les classes enfants de la classe C_i
$\Uparrow (C_i)$	L'ensemble de tous les ancêtres de la classe C_i
$\Downarrow (C_i)$	L'ensemble de tous les descendants de la classe C_i
$\leftrightarrow (C_i)$	L'ensemble des classes sœurs de la classe C_i
$*(C_i)$	Les instances dont la classe la plus spécifique est C_i
\	Désigne la différence,

La politique exclusive

$Tr^{+}(C_i) = *(C_i)$ et $Tr^{-}(C_i) = Tr \setminus * (C_i)$

Les instances positives sont seulement celles étiquetée explicitement par C_i, tandis que les autres sont considérées négatives. Par exemple, $Tr^{+}(C_{2.1})$ comprend les instances dont la classe la plus spécifique est {2.1}, et $Tr^{-}(C_{2.1})$ se compose des instances ayants les classes {1,1.1,1.2,2,2.1.1,2.1.2,2.2,2.2.2,2.2.1}. Les limites de cette méthode est qu'elle ne prend pas en compte la structure hiérarchique lors de la création de l'ensemble d'apprentissage local.

La politique du moins inclusive :

$Tr^{+}(C_i) = \{*(C_i) \cup (\Downarrow (C_i))\}$ et $Tr^{-}(C_i) = Tr \setminus \{*(C_i) \cup (\Downarrow (C_i))\}$.

L'ensemble des instances positives inclut également les instances des classes descendantes, tandis que le reste des instances sont étiquetées négatives. Ainsi, $Tr^{+}(C_{2.1})$ se compose des instances ayants les classes {2.1,2.1.1,2.1.2} et $Tr^{-}(C_{2.1})$ se compose des instances ayants les classe {1,1.1,1.2,2,2.2,2.2.2,2.2.1}.

La politique inclusive :

$Tr^{+}(C_i) = \{*(C_i) \cup \Downarrow (C_i)\}$ et $Tr^{-}(C_i) = Tr \setminus \{*(C_i) \cup \Downarrow (C_i) \cup \Uparrow (C_i)\}$.

Les instances positives de cette politique sont obtenues de la même façon que dans la politique précédente, mais les instances négatives sont obtenues en excluant les instances positives avec leurs ancêtres et descendants. $Tr^{+}(C_{2.1})$ se compose des instances ayants les classes {2.1,2.1.1,2.1.2} et $Tr^{-}(C_{2.1})$ se compose des instances ayants les classes {1,1.1,1.2,2.2,2.2.2,2.2.1}. Il n'inclut pas la classe C_2.

La politique sibling :

$Tr^{+}(C_i) = \{*(C_i) \cup \Downarrow (C_i)\}$ et $Tr^{-}(C_i) = Tr \backslash \{*(C_i) \cup \Downarrow (\leftrightarrow (C_i))\}$.

La difference avec la politique précédente est que l'ensemble des instances négatives contient ayant les classes soeurs de C_i et de leurs descendants. $Tr^{+}(C_{2.1})$ se compose des instances ayants les classes {2.1,2.1.1,2.1.2} et $Tr^{-}(C_{2.1})$ se compose des instances ayants les classes {2.2,2.2.2,2.2.1}.

La politique sibling exclusive (frères et sœurs exclusifs) :

$Tr^{+}(C_i) = *(C_i)$ et $Tr^{-}(C_i) = (\leftrightarrow (C_i))$.

Les instances positives sont seulement celles étiquetées explicitement avec C_i alors que les instances négatives sont celles étiquetées par les classes sœurs de C_i. $Tr^{+}(C_{2.1})$ se compose des instances ayants les classes {2.1} et $Tr^{-}(C_{2.1})$ se compose des instances ayants les classes {2.2}.

2.2.2. Phase d'apprentissage :

Les informations locales peuvent être utilisées de différentes manières, selon la façon dont les classifieurs locaux sont induits. En gros, trois stratégies principales pour l'utilisation des informations locales ont été identifiées **[40]** : un classificateur local par nœud (LCN), un classifieur local par nœud parent (LCPN) et un classificateur local par niveau (LCL). La stratégie LCN forme un classificateur binaire pour chaque classe de la hiérarchie. La stratégie LCPN forme, pour chaque classe interne, un classifieur multi-classe pour distinguer ses sous-classes, et la stratégie LCL forme un classificateur multi-classe pour chaque niveau hiérarchique, où chaque classifieur est responsable de la prédiction dans son associé niveau.

- **Classifieur Local par nœud (LCN) :**

  C'est l'approche la plus utilisée. Pour chaque nœud de la hiérarchie (à l'exception du nœud racine), un classifieur binaire est induit pour prédire si une instance appartient ou pas à la classe. Dans la Figure 26, dix classifieurs sont induits (formés), ce qui est égale au nombre de classes dans la hiérarchie.

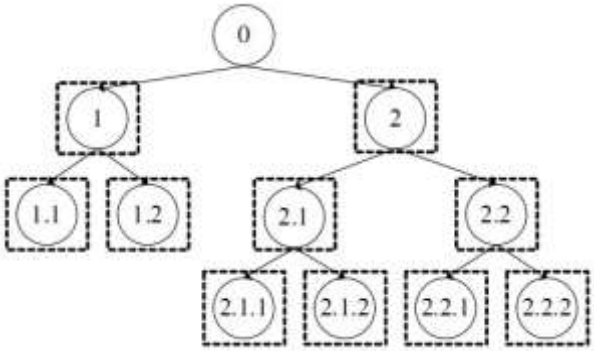


Figure 26 : Classifieur local par nœud, chaque cercle représente une classe et chaque réctangle un classifieur binaire

- **Classifieur local par noeud parent (LCPN) :**

  Cette approche généré un nombre de classifieurs multi-label égale au nombre de classes parentes (noeuds parents) dans la hiérarchie. Le nombre de sortie pour chaque classifieur multi-label est égale au nombre de ses noeuds enfants. La Figure 27 illustre cette approche. Le classifieur au niveau du nœud racine doit être construit.

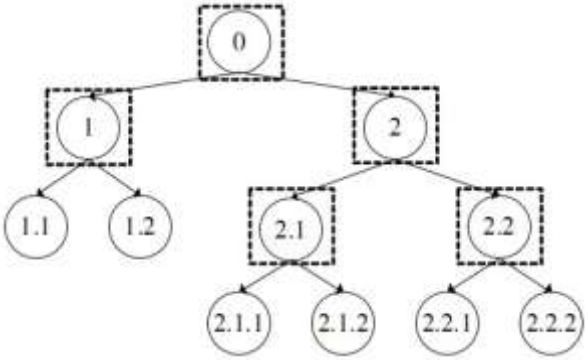


Figure 27: Classifieur local par noeud parent, les réctangles représentent les classifieurs multi-label

- **Classifieur local par niveau (LCL) :**

  Cette approche est couramment utilisée dans une hiérarchie de classes arborescente. Pour chaque niveau de la hiérarchie, un classifieur multilabel est induit comme dans LCPN. La Figure 28 montre un exemple de cette approche, il y a trois classifieurs induits.

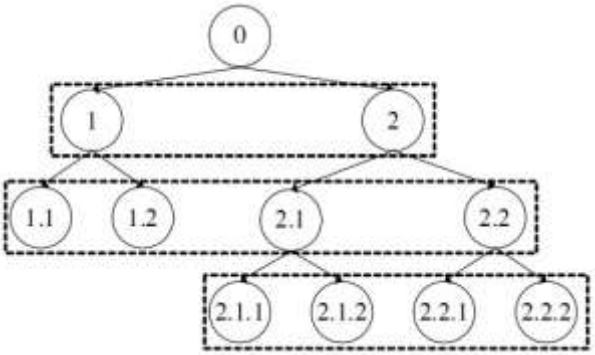


Figure 28: Classifieur local par niveau, les rectangles représentent les classifieurs multi-label

2.2.3. Phase de test :

On commence au nœud racine, une instance est classifiée de manière descendante (Top-down). Lorsqu'elle est affectée à une classe, l'instance sera soumise à un nouveau classifieur afin de prédire à qui de ses sous classes elle appartient. Cette procédure est répétée jusqu'à ce qu'on atteint une classe des noeuds feuilles. Ou jusqu'à ce qu'aucune prédiction supplémentaire n'est possible. Reprenons le système représenté par la Figure 26 où chaque classe a son propre classifieur. Supposons qu'une instance inconnue est prédite dans C_2 mais pas dans $C_{2.2}$. Ainsi cette instance n'a pas besoin de continuer le processus de classification de niveau inférieur {2.2.1,2.2.2}.

En prenant en considération la classification multi-label, une instance inconnue (nouvelle instance) peut être simultanément prédite par plusieurs classifieurs. Par exemple, supposons qu'une instance est assignée à la fois aux classes $C_{1.1}$ et $C_{2.2.1}$. Par conséquent, tous les classifieurs le long des sous arbres de C_1 et C_2 doivent être utilisés pour faire la prédiction de cette instance.

2.3. Approche de classification globale (Big bang) :

Dans l'approche globale, comme son nom l'indique, l'objectif est d'apprendre un modèle global unique pour toutes les classes de la hiérarchie **[35,37]**. Comme le montre la Figure 29.

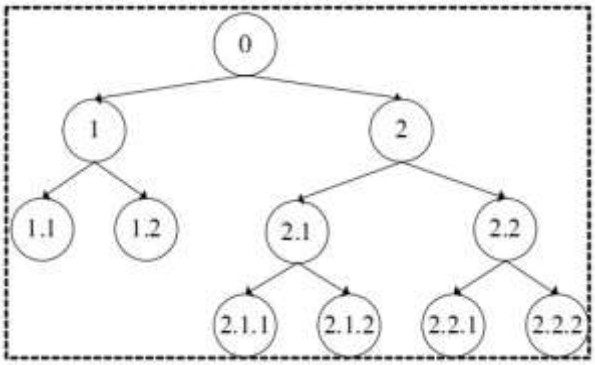


Figure 29: Approche de classification globale, le rectangle représente l'unique classifieur utilisé pour toute la hiérarchie de classes

3. Métriques d'évaluation :

Dans les domaines avec des classes organisées sous forme d'une hiérarchie, les mesures classiques d'évaluation comme la précision, le rappel et la mesure F semblent être insuffisantes pour répondre à la nature hiérarchique des classes. Dans **[43]**, les auteurs proposent une version étendue de ces trois métriques. Dans le contexte hiérarchique, chaque instance appartient non seulement à une (ou plusieurs) classe mais aussi à tous ses ancêtres dans la hiérarchie, à l'exception du nœud racine.

Les métriques pour la i-ième instance sont résumées dans la Table 2, où P_i et T_i représentent les ensembles des classes prédites par le classifieur et des vraies classes respectivement. $\hat{P}_i$ et $\widehat{T\imath}$ représentent P_i et T_i ainsi que leurs ancêtres :

1. La précision hiérarchique (hP_{ri}) est l'exactitude du chemin de la prédiction.
2. Le rappel hiérarchique (hR_i) est la précision du vrai chemin.
3. La mesure F hiérarchique est la combinaison de (hP_{ri}) et (hR_i).

Precision	*Rappel*	Mesure *h*F
$hPri = \frac{\hat{P}i \cap \hat{T}i}{\hat{P}i}$	$hRri = \frac{\hat{P}i \cap \hat{T}i}{\hat{T}i}$	$F = \frac{2 \times hPri \times hRri}{hPri + hRri}$

Table 2 : Métriques d'évaluation hiérarchique pour une seule instance x_i

Exemple :

En se référant à la structure hiérarchique de la figure 30

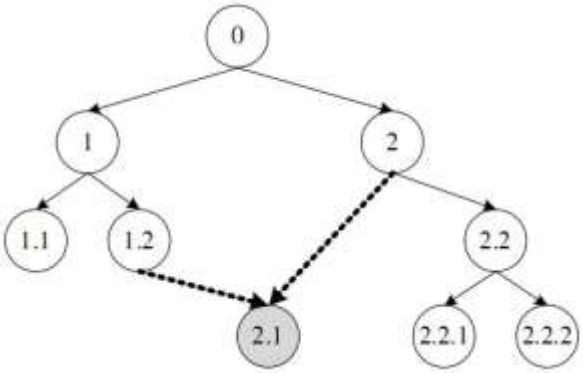


Figure 30 : Exemple d'une structure hiérarchique en DAG

Supposons que $P_i = \{C_{2.2}\}$, $T_i = \{C_{2.2.1}\}$ et donc, $\hat{P}_i = \{C_2, C_{2.2}\}$, $\widehat{Ti} = \{C_2, C_{2.2}, C_{2.2.1}\}$:

- La précision hiérarchique $hPri = \frac{2}{2} = 1$
- Le rappel hiérarchique $hRi = \frac{2}{3}$
- La valeur de la mesure $\mathcal{NX}$ hiérarchique, *h*F = 0,8

Micro-averaging « micro-moyenne » :

La performance hiérarchique pour un ensemble avec *n* instances étiquetées par *l* labels est présenté dans la Table 3.

Precision hP_r^u	*Rappel* hR_e^u	Mesure hF^u
$hP_r^u = \frac{\sum_{i=1}^{n} \hat{P}i \cap \hat{T}i}{\sum_{i=1}^{n} \hat{P}i}$	$hR_e^u = \frac{\sum_{i=1}^{n} \hat{P}i \cap \hat{T}i}{\sum_{i=1}^{n} \hat{T}i}$	$hF^u = \frac{2 \times hPri \times hRi}{hPri + hRi}$

Table 3 : Métriques d'évaluation hiérarchique pour un ensemble de *n* instances

Exemple :

Considérons l'ensemble d'apprentissage S = {(x_1, $C_{2.2}$), (x_2, C_1), (x_3, C_2)} et que les classes sont organisées selon la hiérarchie de la Figure 29. Supposons qu'ils y aient deux classifieurs Φ_1 et Φ_2. Les résultats de leurs prédictions sur l'ensemble d'apprentissage S sont :

$$\Phi_1(S) = \{(x_1, C_{2.2}), (x_2, C_2), (x_3, C_1)\}$$
$$\Phi_2(S) = \{(x_1, C_{1.1}), (x_2, C_1), (x_3, C_2)\}$$

L'évaluation hiérarchique pour chaque instance des deux classifieurs Φ_1 et Φ_2 sur l'ensemble d'apprentissage S est donnée par la Table 4 et la Table 5, respectivement.

Instances	Précision (hP_{ri})	Rappel (hR_i)	Mesure F
x_1	$\frac{2}{2}$=1	$\frac{2}{2}$=1	1
x_2	$\frac{0}{1} = 0$	$\frac{0}{1} = 0$	0
x_3	$\frac{0}{1} = 0$	$\frac{0}{1}= 0$	0

Table 4: Performances hiérarchique, pour chaque instance, du classifieur Φ_1

Instances	Précision (hP_{ri})	Rappel (hR_i)	Mesure F
x_1	$\frac{0}{2} = 0$	$\frac{0}{2} = 0$	0
x_2	$\frac{1}{1}$ =1	$\frac{1}{1}$ =1	1
x_3	$\frac{1}{1}$ =1	$\frac{1}{1}$ =1	1

Table 5: Performances hiérarchique, pour chaque instance, du classifieur Φ_2

On remarque que les performances de prédiction du classifieur Φ_2 sont meilleures que celles de Φ_1. Mais l'évaluation pour l'ensemble de toutes les instances (micro-averaging) indique que les performances des deux classifieurs sont les mêmes $hP_r^u = hR_e^u = hF^u = 1/2$. Donc les résultats de cette méthode peuvent être biaisés. Pour évaluer raisonnablement les classifieurs dans le scénario précédent, on utilise l'évaluation macro-averaging (macro-moyenne) qui fusionne les différentes métriques pour toutes les instances, comme le montre la table 6.

Precision hP_r^M	*Rappel* hR_e^M	Mesure hF^M
$hP_r^M = \frac{\sum_{i=1}^{n} hPi}{n}$	$hR_e^M = \frac{\sum_{i=1}^{n} hRi}{n}$	$hF^M = \frac{\sum_{i=1}^{n} hFi}{n}$

Table 6 : Métrique d'évaluation hiérarchique macro-averaging

- $\Phi_{1(S)}$: $hP_r^M = hR_{e=}^M hF^M = \frac{1}{3}$
- $\Phi_{2(S)}$: $hP_r^M = hR_{e=}^M hF^M = \frac{2}{3}$

Cette méthode montre que les performances du classifieur Φ_2 sont meilleures que celles de Φ_1.

Conclusion :

Dans ce chapitre, nous avons présenté quelques notions relatives à la classification multi label hiérarchique, nous avons, particulièrement, constaté que ce type de classification nous permet de tenir compte des relations qui existent entre les classes et permet aussi de structurer les classes sous une forme hiérarchique, arborescente ou DAG. Les méthodes de classification hiérarchiques peuvent être groupées en trois approches : la classification hiérarchique plate, la classification hiérarchique locale et la classification globale.

Chapitre 3

Analyse de concepts formels

1.Introduction :

De nombreuses méthodes ont été développées et regroupées sous le nom de la "fouille de Données". L'idée est de fouiller dans une masse de données pour en extraire de nouvelles informations sur les liaisons existantes entre des sous-ensembles de données. Ceci peut servir à réutiliser les données pour d'autres besoins et à analyser des données pour découvrir des phénomènes inconnus. Dans plusieurs domaines, les observations sont disponibles mais l'utilisateur ne sait pas comment les utiliser.

L'analyse formelle de concepts est une méthode ensembliste permettant de générer et de structurer un ensemble de concepts à partir des relations entre des objets et des propriétés.

2. Notions de Base :

Nous allons, dans cette section, rappeler des définitions et des concepts mathématiques nécessaires à l'introduction de l'ACF.

Définition 2.1. (**Relation Binaire)** :

Une relation binaire R entre deux ensembles arbitraires X et Y est définie dans le produit Cartésien $X \times Y$, elle consiste en un ensemble de paires $(x ; y)$ avec $x \in X$ et $y \in Y$. On note xRy quand $(x ; y) \in R$.

Quand X = Y, R est dite relation binaire sur X.

Définition 2.2. (Relation d'ordre) :

La relation binaire R sur un ensemble E est dite relation d'ordre si elle satisfait les conditions suivantes :

1. (Réflexive) $\forall x \in E : xRx$;

2. (Antisymétrique) $\forall x, y \in E : xRy \wedge yRx \Rightarrow x = y$;
3. (Transitive) $\forall x, y, z \in E : xRy \wedge yRz \Rightarrow xRz$.

La relation d'ordre R est dite totale si : $\forall x, y \in E, xRy \vee yRx$, autrement dite partielle.

Nous utilisons souvent la notation $\leq$ pour désigner une relation d'ordre.

Définition 2.3. (Ensemble ordonné) :

On appelle ensemble ordonné tout couple $(\varepsilon, \leq)$ formé d'un ensemble ε et d'une relation d'ordre $\leq$ sur ε.

Définition 2.4. (Supremum, Infimum) :

Soit $(\varepsilon, \leq)$ un ensemble ordonné et E un sous ensemble de ε (i.e. $E \subseteq \varepsilon$).

L'élément s est appelé borne supérieure ou supremum de E (noté $\bigvee E$) si $\forall e \in E$, $e \leq s$. Duallement, l´élément i est appelé borne inferieure ou infimum de E (noté $\bigwedge E$) si $\forall e \in E$, $i \leq e$.

Dans tout ensemble ordonné, lorsque le supremum (respectivement l'infimum) existe, il est unique.

Définition 2.5. (Treillis) :

Un treillis est un ensemble partiellement ordonné $(\varepsilon, \leq)$ tel que $x \vee y$ et $x \wedge y$ existent pour tout couple d´éléments $x, y \in \varepsilon$.

Un treillis est dit complet si $\bigvee E$ et $\bigwedge E$ existent pour tout sous ensemble E de ε.

En particulier, un treillis complet admet un élément maximal (top) noté par $\top$et un élément minimal (bottom) noté par $\bot$.

Tout treillis fini est un treillis complet.

Définition 2.6 (Demi-treillis) : Un ensemble ordonné $(\varepsilon, \leq)$ est un sup-demi-treillis (respectivement Inf-demi-treillis) si tout couple d'éléments $x, y \in \varepsilon$ admet un supremum $x \vee y$ (respectivement un infimum $x \wedge y$).

Définition 2.7. (Fermeture) :

On appelle opérateur de fermeture sur un ensemble ordonné $(X, \leq)$ toute application $\Phi : X \rightarrow X$ vérifiant les propriétés suivantes :

- Φ est extensive : $\forall x \in X, x \leq \Phi(x)$;
- Φ est isotone : $\forall x, y \in X, x \leq y \Rightarrow \Phi(x) \leq \Phi(y)$;
- Φ est idempotent : $\forall x \in X, \Phi(x) = \Phi(\Phi(x))$.

Définition 2.8. (Ouverture) :

On appelle operateur d'ouverture sur un ensemble ordonné $(X, \leq)$ toute application $\Psi : X \rightarrow X$ vérifiant les propriétés suivantes :

- Ψ est contractante : $\forall x \in X, \Psi(x) \leq x$;
- Ψ est isotone : $\forall x, y \in X, x \leq y \Rightarrow \Psi(x) \leq \Psi(y)$;
- Ψ est idempotent : $\forall x \in X, \Psi(x) = \Psi(\Psi(x))$;

Définition 2.9. (Connexion de Galois) :

Soient $\Phi : A \rightarrow B$ et $\Psi : B \rightarrow A$ deux applications définies sur les ensembles ordonnés $(A, \leq_A)$ et $(B, \leq_B)$. Φ et Ψ forment une connexion de Galois (ou une correspondance de Galois) entre $(A, \leq_A)$ et $(B, \leq_B)$ si elles vérifient les conditions suivantes :

1. $\forall a_1, a_2 \in A, a_1 \leq_A a_2 \Rightarrow \Phi(a_2) \leq_B \Phi(a_1)$;
2. $\forall b_1, b_2 \in B, b_1 \leq_B b_2 \Rightarrow \Psi(b_2) \leq_A \Psi(b_1)$;
3. $\forall a \in A, \forall b \in B, a \leq_A \Psi(\Phi(a)) \wedge b \leq_B \Phi(\Psi(b))$.

3.Analyse de Concepts Formels :

L'Analyse Formelle de Concepts (AFC) (appelé aussi Analyse de Concepts Formels (ACF)) **[44, 45]** est un formalisme mathématique pour l'analyse de données, la représentation de connaissances et la visualisation de connaissances. L'idée de base de l'AFC est d'extraire des concepts regroupant des objets et leurs propriétés/attributs à partir de données et de construire une hiérarchie à partir de ces concepts. Elle a trouvé usage dans plusieurs domaines comme la psychologie, la sociologie, l'anthropologie, la médecine, la biologie, la linguistique, l'informatique et les mathématiques **[46]**

L'Analyse de Concepts Formels (ACF) a été introduite par le Wille **[47]** en tant que méthode de représentation de connaissances et d'analyse de données. Les fondements théoriques de l'ACF reposent sur la théorie des treillis définis par G. Birkho et la théorie des ensembles. Wille a utilisé l'interprétation philosophique du concept comme une unité de pensée

comprenant un ensemble d'objets et un ensemble de leurs attributs communs. Par la suite, l'ACF a été consolidée mathématiquement par Ganter et Wille **[48]**.

L'ACF repose sur la notion de "Contexte Formel" qui représente une relation binaire reliant un ensemble d'objets à un ensemble d'attributs (propriétés). Généralement un contexte formel est représenté par une matrice avec en ligne les objets et en Colonne les attributs. Une marque placée sur l'intersection d'une ligne et d'une colonne signe que l'objet de la ligne possède l'attribut de la colonne. L'ACF consiste à induire des paires de sous-ensembles d'objets et d'attributs le sous-ensemble Objets est appelé extension et le sous-ensemble Attributs est appelé intension. Pour un concept formel donné, les objets de l'extension possèdent tous les attributs de son intention, De même tous les attributs de l'intention sont partagés par tous les objets de son extension.

L'ACF permet de définir une hiérarchie dite subsomption entre les concepts formels qui peut être interprétée comme une relation de généralisation/spécialisation entre les concepts formels. Tous les concepts formels d'un contexte formel ainsi que toutes leurs relations peuvent être transformés en une structure de treillis de concepts qui peut être présentée sous forme de diagramme de Hasse. Mathématiquement, les concepts formels sont induits en utilisant l'opérateur de dérivation de Galois, appelé également opérateur de suffisance **[46, 49]**. Cet opérateur forme une connexion de Galois entre les sous-ensembles d'objets et d'attributs et sa composition forme un opérateur de fermeture.

3.1 Contexte formel :

Définition 3.1.1 (Contexte formel) : Un contexte formel est un triplet (G, M, R) ou G est un ensemble d'objets, M est un ensemble de propriétés/attributs et R est une relation binaire entre G et M. Un couple $(g, m) \in R$ (également noté gRm) signifie que l'objet $g \in G$ possède la propriété $m \in M$.

Un contexte formel est représenté sous la forme d'un tableau à deux dimensions où les lignes correspondent aux objets et les colonnes aux attributs. Les cases du tableau sont remplies suivant la présence/absence de la propriété, autrement dit si le $i^{éme}$ objet g est en relation R avec le $j^{éme}$ alors la case l'intersection de la ligne i et de la colonne j contient "×", sinon la case est vide.

R	Ovipare	Denté	Volant	Aquatique	Mammifère
Lièvre		×			×
Chauve-souris		×	×		×
Canard	×		×	×	
Castor		×		×	×
Tortue	×			×	
Crocodile	×	×		×	

La table 7: exemple de contexte formel représentant un ensemble d'animaux et certains de leurs attributs (propriétés).

3.2. Connexion de Galois dans un contexte formel :

Définition 2 :

Soit (G, M, R) un contexte formel. Pour tout $A \subseteq G$ et $B \subseteq M$, on définit :

$$A' = \{m \in M | \forall g \in A, gRm\}$$
$$B' = \{g \in G | \forall m \in B, gRm\}$$

Intuitivement, A' est l'ensemble des attributs communs à tous les objets de A et B' est l'ensemble des objets possédant tous les attributs de B. Les applications $(.)' : 2^G \rightarrow 2^M$ et $(.)' : 2^M \rightarrow 2^G$ sont appelées opérateurs de dérivation entre l'ensemble de parties de G noté par 2^G et l'ensemble de parties de M noté par 2^M. Ils sont également appelés opérateurs de suffisance dans **[49]**

La composition de ces opérateurs produit deux opérateurs $(.)'' : 2^G \rightarrow 2^G$ et $(.)'' : 2^M \rightarrow 2^M$. Le premier opérateur permet d'associer un ensemble d'objets A l'ensemble maximal d'objets dans G ayant les attributs communs aux objets de A. Cet ensemble est noté A**''**. De façon duale, le second opérateur permet d'associer un ensemble d'attributs B l'ensemble maximal d'attributs dans M communs aux objets ayant les attributs dans B. Cet ensemble est noté B**''**.

Les opérateurs $(.)'' : 2^G \rightarrow 2^G$ et $(.)'' : 2^M \rightarrow 2^M$ définissent deux fermetures, respectivement sur l'ensemble des parties de G et sur l'ensemble des parties de M. Les ensembles A**''** et B**''** sont des fermés pour ces deux opérateurs respectifs. L'ensemble des fermés de 2^G muni de l'inclusion est un treillis complet. De la même façon, l'ensemble des fermés de 2^M muni de l'inclusion est un treillis complet.

Exemple 3. En considérant le contexte formel du Tableau 7, nous donnons ci-dessous, quelques exemples de l'application de l'opérateur de suffisance ainsi que sur sa composition :

- {Lièvre, Chauve-souris} **'** = {Denté, Mammifère}
- {Lièvre, Chauve-souris, Canard, Castor, Tortue, Crocodile} **'** = ∅
- {Ovipare, Denté} **'** = {Crocodile}
- {Ovipare, Denté, Volant, Aquatique, Mammifère} **'** = ∅
- {Lièvre, Chauve-souris} **"** = {Lièvre, Chauve-souris, Castor}
- {Ovipare, Denté} **"** = {Ovipare, Denté, Aquatique

3.3. Concept formel :

Après avoir défini un contexte formel et l'opérateur de suffisance, nous donnons ci-après la définition d'un concept formel. Soit (G, M, R) un contexte formel. Un concept formel est un couple (A, B) tel que A ⊆ G, B ⊆ M, A**'** = B et B**'** = A. A et B sont respectivement appelés extension (extent) et intension (intent) du concept formel (A, B).

Dans un contexte formel, un concept correspond un rectangle maximal de la table formée par la relation binaire du contexte : tout objet de l'extension à tous les attributs de l'intention. Il est important de noter que cette notion de rectangle maximal est indépendante de l'ordre des lignes et des colonnes. Ces ensembles maximaux d'objets et d'attributs correspondent à des fermés dans 2^G et 2^M respectivement. Un sous-ensemble B de M est l'intention d'un concept formel dans (G, M, R) si et seulement si B**"** = B (B est fermé pour (.)**")** et, de façon duale, un sous ensemble A de G est l'extension d'un concept formel dans l'ensemble de concepts du contexte (G, M, R) si et seulement si A**"** = A (A est fermé pour (.)**"**).

Exemple 4. En considérant le contexte formel du Tableau 7, nous donnons ci-dessous, quelques exemples de paires A, B qui sont (ou non) des concepts formels :

- {Lièvre, Chauve-souris, Castor}, {Denté, Mammifère} est un concept formel,
- {Lièvre, Chauve-souris, Canard, Castor, Tortue, Crocodile}, est un concept formel,
- {Canard, Crocodile}, {Ovipare, Dente} n'est pas un concept formel.

Il est aussi important de préciser que les concepts du treillis du contexte (G, M, R) sont ordonnés par une relation d'ordre hiérarchique entre concepts (appelé aussi relation de subsomption) noté " ≤ " et définie comme suit :

Définition 3 (Relation de "subsomption") :

Soient (A_1, B_1) et (A_2, B_2) deux concepts formels de Ɓ (G, M, R).

$(A_1, B_1) \leq (A_2, B_2)$ si et seulement si $A_1 \subseteq A_2$ (ou de façon duale $B_2 \subseteq B_1$).

(A_2, B_2) est dit **super-concept** de (A_1, B_1) et (A_1, B_1) est dit **sous-concept** de (A_2, B_2).

La relation " $\leq$ " est dite **relation de subsomption**.

Exemple 5 : À partir du contexte formel du Tableau 7, nous avons :

- {Chauve-souris}, {Dente, Volant, Mammifère} est un sub-concept de {Lièvre, Chauve-souris, Castor}, {Dente, Mammifère}.
- {Lièvre, Chauve-souris, Castor}, {Denté, Mammifère} est un super-concept de {Chauve-souris}, {Denté, Volant, Mammifère}.

La relation " $\leq$ " s'appuie sur deux inclusions duales, entre ensembles d'objets et entre ensembles d'attributs et peut ainsi être interprété comme une relation de généralisation/spécialisation entre les concepts formels. Un concept est plus général qu'un autre concept s'il contient plus d'objets dans son extension. En contre partie, les attributs partagés par ces objets sont réduits. De façon duale, un concept est plus spécifique qu'un autre s'il contient moins d'objets dans son extension. Ces objets ont plus d'attributs en commun.

3.4.Treillis de concepts :

Définition 4 (Treillis de concepts) :

La relation " $\leq$ " permet d'organiser les concepts formels en un treillis complet ($\mathfrak{B}$ (G, M, R), $\leq$) appelé treillis de concepts ou encore treillis de Galois **[50]** et noté par $\mathfrak{B}$ (G, M, R) ou $\mathfrak{B}$ (K). Le supremum et l'infimum dans $\mathfrak{B}$ (K) sont donnés par :

$$\bigwedge_{j\in J}(A_j, B_j) = \left(\bigcap_{j\in J} A_j, \left(\bigcup_{j\in J} B_j\right)''\right)$$

$$\bigvee_{j\in J}(A_j, B_j) = \left(\left(\bigcup_{j\in J} A_j\right)'', \bigcap_{j\in J} B_j\right)$$

Un treillis de concepts formels est une représentation équivalente à un contexte formel qui met en avant les groupements possibles entre objets et propriétés ainsi que les relations d'inclusion entre ces groupements.

La Figure 31 : illustre le treillis de concepts formels associé à la Table 7. Les objets Lièvre, Chauve-souris, Canard, Castor, Tortue et Crocodile sont désignés respectivement. Par les objets o_1, o_2, o_3, o_4, o_5 et o_6. De même, les attributs Ovipare, Denté, Volant, Aquatique et Mammifère sont désignés respectivement. Par a_1, a_2, a_3, a_4 et a_5.

R	a_1	a_2	a_3	a_4	a_5
o_1		×			×
o_2		×	×		×
o_3	×		×	×	
o_4		×		×	×
o_5	×			×	
o_6	×	×		×	

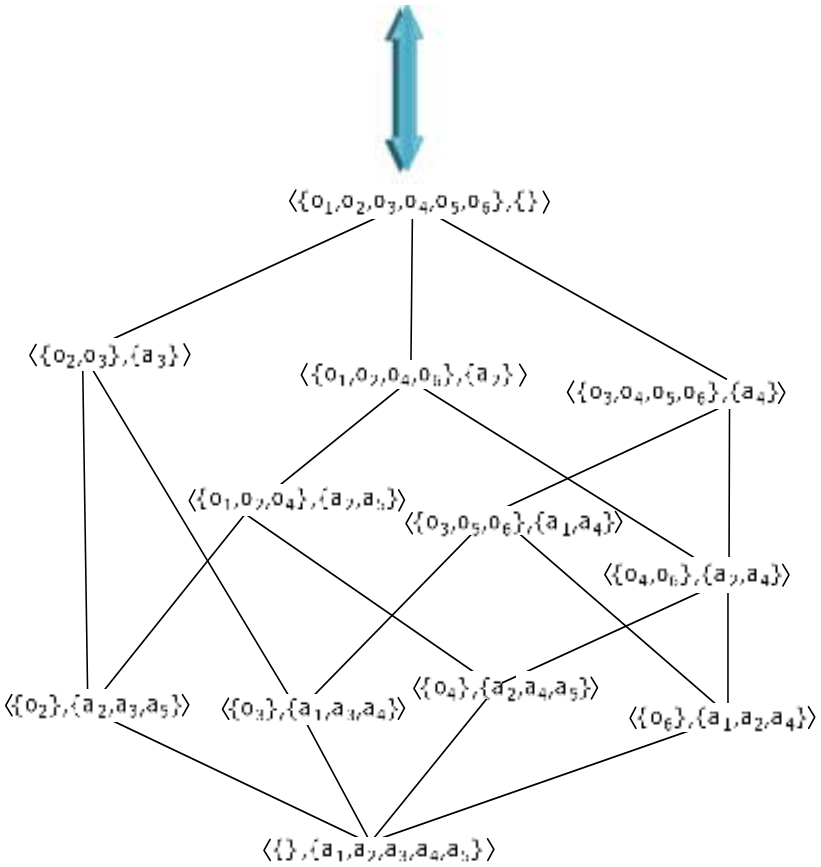


Figure 31 : Diagramme de Hasse du Treillis de Concepts Formels Associé au Contexte de la table 7.

Dans ce diagramme (appelé diagramme de Hasse) chaque nœud dénote un concept formel, quant aux ligne du diagramme, elles dénotent les relations directes entre les nœuds (les concepts formels). La lecture descendante de ces lignes correspond à la relation de super-concept. Duallement la lecture ascendante correspond à la relation de sub concept. Notons que le treillis de concepts formels est isomorphe au contexte formel. De plus, la représentation graphique du

treillis sous forme de diagramme de HASSE, facilite la compréhension et l'interprétation de la relation de subsomption entre les concepts formels ainsi que les relations entre les objets et les attributs au sein du même concept formel.

La construction du treillis de concepts d'un contexte formel peut être décomposée en deux parties. La première concerne le calcul des concepts formels, et la deuxième vise la représentation graphique du treillis ou la construction du diagramme de HASSE correspondant à ce treillis **[50].**

Le problème de calcul des concepts formels à partir d'un contexte formel a fait l'objet de nombreux travaux de recherche, ainsi, plusieurs algorithmes ont été proposés dans la littérature. Parmi ces travaux, nous citons : Chein **[51]**, Norris **[52]**, NextClosure **[53 ; 54]**, Bordat **[55],** Close-by-One **[56]**, Godin **[57]**, Galois **[58],** Nourine **[59]**,Titanic **[60,61]**, Divide&Conquer [**62,63**] et AddIntent **[64].**

5.Algorithmes de construction de treillis de concepts :

La construction du treillis de concepts d'une relation binaire donnée peut être décomposée en trois parties **[67,68]** :

1. L'énumération des rectangles maximaux (les fermés),
2. La recherche de la relation d'ordre partiel entre ces rectangles
3. Et la représentation graphique du treillis (construction du diagramme de HASSE correspondant au treillis).

Les deux premières étapes constituent le problème de calcul des concepts d'un treillis de concepts à partir d'un contexte formel et peuvent être exécutées simultanément ou de manière séquentielle (dans l'ordre donné plus haut). Cependant, la troisième étape fait partie du problème de visualisation de graphes. De ce fait, ces deux problématiques sont souvent traitées indépendamment ce qui a donné lieu à deux axes de recherche complémentaires : le premier consiste à proposer des algorithmes de plus en plus performants (complexité, temps d'exécution, occupation mémoire, passage à l'échelle) pour le calcul de treillis de concepts à partir de contextes formels et le deuxième consiste à fournir des outils efficaces pour la visualisation de ces treillis.

Le problème de calcul des concepts d'un treillis de concepts à partir d'un contexte formel a fait l'objet de nombreux travaux de recherche **[70, 71,73,74]** qui ont abouti à la proposition d'une variété d'algorithmes dont les plus connus sont Chein **[51]**, Norris **[52]** (la méthode dite de

malgrange), NextClosure **[53, 75]**, Bordat (**[55]**, Close-by-One **[69]**, Godin **[76]**, Galois **[77**, **78]**, Nourine **[59]**, Lindig **[72]**, Titanic **[60, 61]**, Divide&Conquer **[62, 63]** et AddIntent **[64]**. Chacun de ces algorithmes se distingue des autres par plusieurs critères dont la stratégie de calcul des concepts, la recherche de l'ordre entre ces concepts, les structures de données utilisées pour le stockage des résultats intermédiaires et le résultat final. Les principaux algorithmes ont fait l'objet d'une comparaison détaillée dans **[79]**. Cette comparaison a montré qu'aucun algorithme n'est meilleur que tous les autres sur tous les plans et que les performances d'un algorithme dépendent fortement des caractéristiques du contexte formel en entrée.

Dans la suite de cette section nous allons répartir les algorithmes en fonction de leurs stratégies d'acquisition de données à partir d'un contexte formel. Considérant ce critère, on distingue trois familles d'algorithmes : les algorithmes batch qui considèrent la totalité du contexte dès le départ, les algorithmes incrémentaux qui considèrent le contexte ligne par ligne et les algorithmes d'assemblage qui répartissent le contexte en deux et calculent les concepts correspondant à chaque moitié puis font l'assemblage.

5.1 Algorithmes batch :

Les algorithmes batch constituent la première génération des algorithmes de construction de treillis. Ils prennent en entrée le contexte formel tout entier et calculent les concepts formels et l'ordre entre ces concepts simultanément ou de manière séquentielle.

L'un des premiers algorithmes proposés dans cette catégorie est l'algorithme de Chein **[51]** qui génère les concepts par niveaux. L'algorithme est itératif, Son point de départ est l'ensemble L_1 de couples (A, B) représentant les lignes du contexte formel (A contient un seul élément de G et $B = A'$). A chaque étape i, l'algorithme part d'un ensemble L_i et construit les éléments de L_{i+1}. Un élément ($A3$, $B3$) de L_{i+1} est obtenu en combinant deux éléments ($A1$, $B1$) et ($A2$, $B2$) de L_i comme suit : $A3 = A1 \cup A2$ et $B3 = B1 \cap B2$. Les éléments de L_i inclus dans au moins un élément de L_{i-1} ne sont pas maximaux et sont donc supprimés. L'algorithme s'arrête lorsque L_{i+1} contient moins de deux éléments. Les éléments non supprimés après l'arrêt de l'algorithme sont les concepts du contexte formel considéré. Pour illustrer le fonctionnement de cet algorithme, un exemple de contexte formel représentant les planètes du système solaire est représenté dans la table ci-dessous. Dans cet exemple, Mercure possède les attributs : petite taille, proche du soleil et n'est pas un satellite.

attribut / Objets	Taille			Distance au soleil		Satellite	
	Petite	moyenne	Grande	Proche	Loin	oui	Non
Mercure	×			×			×
Vénus	×			×			×
Terre	×			×		×	
Mars	×			×		×	
Jupiter			×		×	×	
Saturne			×		×	×	
Uranus		×			×	×	
Neptune		×			×	×	
Pluton	×				×	×	

Table 8: exemple de contexte formel représentant les planètes du système solaire

avec un renommage des attributs et des objets pour faciliter la lisibilité des concepts lors des étapes d'exécution de l'algorithme. Le contexte considéré est donné dans **la table 9** et les traces d'exécution de l'algorithme sont données dans le **la table 10.** Les concepts formels calculés par l'algorithme à partir du contexte formel considéré sont les éléments non barrés dans **la table10**.

	A	B	C	D	e	F	G
1	×			×			×
2	×			×			×
3	×			×		×	
4	×			×		×	
5			×		×	×	
6			×		×	×	
7		×			×	×	
8		×			×	×	
9	×				×	×	

Table 9 : Contexte formel représentant les planètes du système solaire avec renommage des attributs et des objets **.**

L_1	L_2	L_3	L_4
~~1 × adg~~ **(12 × adg)	12 × adg	12349 × a	∅
~~2 × adg~~ **(12 × adg)	~~13 × ad~~ **(134 × ad)	3456789 × f	
~~3 × adf~~ **(34 × adf)	~~134 × ad~~ **(1234 × ad)		
~~4 × adf~~ **(34 × adf)	~~19 × a~~ **(129 × a)		
~~5 × cef~~ **(56 × cef)	1234 × ad		
~~6 × cef~~ **(56 × cef)	~~129 × a~~ ***(12349 × a)		
~~7 × bef~~ **(78 × bef)	34 × adf		
~~8 × bef~~ **(78 × bef)	~~35 × f~~ **(356 × f)		
9 × aef			
	~~356 × f~~ **(3567 × f)		
	~~3567 × f~~ **(35678 × f)		
	~~35678 × f~~ **(345678 × f)		
	~~39 × af~~ **(349 × af)		
	~~345678 × f~~ ***(3456789 × f)		
	~~349 × af~~		
	56 × cef		
	57 ~~× ef~~ **(578 × ef)		
	~~578 × ef~~ **(5789 × ef)		
	~~5789 × ef~~ **(56789 × ef)		
	56789 × ef		
	78 × bef		

Table 10: Les ensembles de rectangles maximaux calculés par l'algorithme de Chein à partir du contexte formel donné dans la table 8.

Les rectangles barrés correspondent à des rectangles non maximaux. Les rectangles qui les contiennent sont indiqués entre parenthèses et l'itération durant laquelle ils ont été calculés est donnée par le nombre de *. Par exemple, "~~1 × adg~~ **(12 × adg)" est interprété comme suit : le rectangle "1 × adg" n'est pas maximal, il est remplacé par
"12 × adg" retrouvé à la deuxième itération de l'algorithme.

5.2 Algorithmes incrémentaux :

Les algorithmes incrémentaux considèrent le contexte formel ligne par ligne (ou colonne par colonne) et construisent le treillis de concepts par ajouts successifs de ligne ou de colonne tout en conservant sa structure. À une étape *k*, les concepts formels correspondants au *k* premières lignes du contexte formel sont calculés. L'ajout de la $(k + 1)^{ème}$ ligne entraîne la modification

d'une partie des concepts calculés à l'étape *k* et l'ajout d'éventuels nouveaux concepts. Conçu ainsi, les algorithmes incrémentaux permettent de gérer les contextes dynamiques, où le nombre d'objets et/ou d'attributs peut évoluer, sans avoir à recalculer le treillis à partir de zéro suite à une modification du contexte. Les étapes de la construction incrémentale du treillis de concepts correspondant au contexte donné dans **la table 9** sont schématisées dans la **figure 32**. Bien que la motivation du coté applicatif des algorithmes incrémentaux n'est apparue qu'au milieu des années 90 **[83, 84, 82]**, le premier algorithme incrémental publié, est l'algorithme de Norris **[52]**, date de 1978 et reste parmi les plus performants **[79]**.

Plus tard, deux nouveaux algorithmes incrémentaux similaires à celui de Norris ont été proposés dans le cadre des approches d'analyse de données par treillis de concepts incluant la classification et la recherche d'information **[84, 58]** : l'algorithme Galois **[81]** et l'algorithme de Godin **[83,85].**

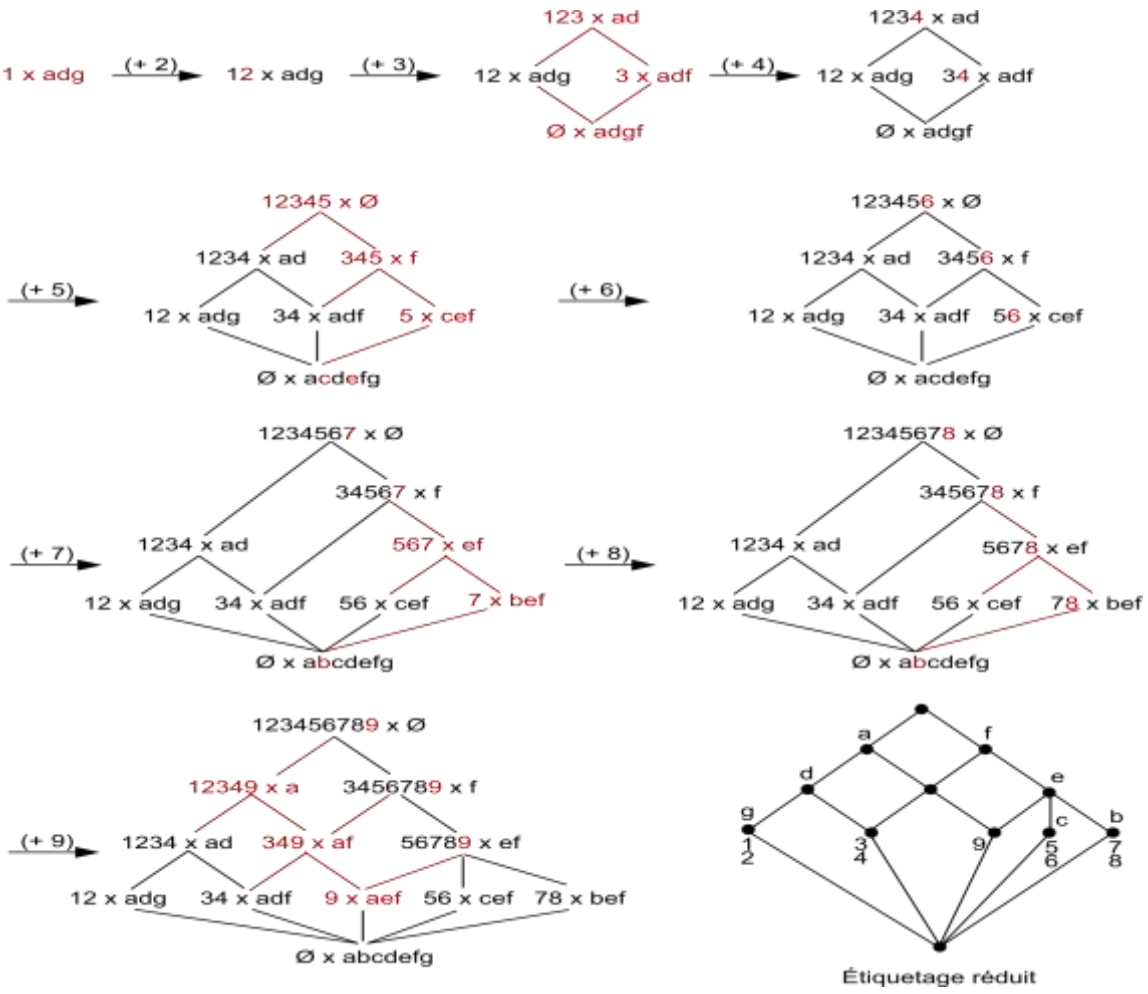


Figure 32: Les étapes de la construction incrémentale du treillis de concepts correspondant au contexte formel donné dans la table 9.

L'ajout d'une nouvelle ligne au treillis construit est symbolisée par " $\xrightarrow{(+n)}$ " où *n* désigne le numéro de la ligne considérée. Les modifications engendrées par un tel ajout sont indiquées en rouge sur le treillis.

5.3 Algorithmes d'assemblage

Les algorithmes d'assemblage constituent une évolution des algorithmes incrémentaux qui généralise le caractère incrémental à des ensembles d'objets/attributs. Ils permettent de diviser un contexte formel en deux parties verticalement ou horizontalement puis de calculer le treillis de concepts correspondant à chaque partie et enfin d'assembler les treillis obtenus en un seul. Le seul algorithme connu de cette famille est l'algorithme Divide&Conquer **[62, 63].** Pour illustrer le principe de cette approche, nous considérons à nouveau le contexte formel K donné dans **la table 9** qu'on divisera en deux contextes K_1 et K_2 .

	a	b	C	D
1	×			×
2	×			×
3	×			×
4	×			×
5			×	
6			×	
7		×		
8		×		
9	×			

	e	f	G
1			×
2			×
3		×	
4		×	
5	×	×	
6	×	×	
7	×	×	
8	×	×	
9	×	×	

Table 11 : Découpage du contexte formel en deux parties. (K_1 à gauche et K_2 à droite).

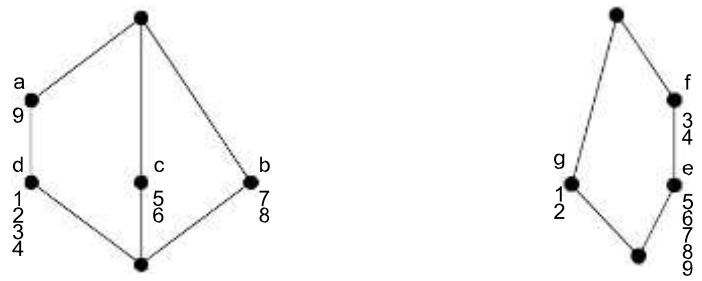


Figure 33: Les treillis de concepts correspondant aux deux contextes K_1 et K_2 formels de la tables 11

6.Implications dans un contexte formel :

Définition 6.1: Soient un contexte formel Ɓ (G, M, R) et $B1, B2 \subseteq M$ deux ensembles d'attributs. On dit que $B1$ implique $B2$ si et seulement tout objet de G qui a les attributs de $B1$ a aussi les attributs de $B2$:

$B1 \rightarrow B2$ ssi $B1' \subseteq B2'$

Dans le contexte formel des planètes du système solaire donné dans la **table 8**, on a l'exemple d'implication suivante :

"Satellite : non" → "Taille : petite", "Distance au soleil : proche"

qui se lit : toute planète n'ayant pas de satellite est de petite taille et proche du soleil. Considérons $B1$ = {"Satellite : non"} et $B2$ = {"Taille : petite", "Distance au soleil : proche"}, nous avons

B1'= {Mercure ,venus} et B2' ={Mercure,Venus,Terre,Mars,Pluton}qui vérifient la définition 6.1 .Une implication de la forme $B1 \rightarrow B2$ peut être ramené à un ensemble d'implication de la forme $B1 \rightarrow b$ pour tout $b \in B2$. L'implication donné en exemple plus haut peut être ramenée aux deux implications suivantes :

"Satellite : non" → "Taille : petite"

et

"Satellite : non" → "Distance au soleil : proche".

7.CONCLUSION :

Dans ce chapitre, nous avons présenté la théorie de l'ACF qui permet de construire un Treillis de Galois à partir d'un contexte formel, ce graphe représente la hiérarchie des caractéristiques des objets. Une fois le treillis construit, l'étape de recherche de points d'intérêt peut être effectuée grâce au mode de classification offert par les treillis de Galois. La méthode permet d'avoir que des points d'intérêt valides ayant au moins l'une des propriétés demandées par l'utilisateur. Le fondement mathématique de l'ACF permet de garantir l'exactitude des réponses délivrées.

Chapitre 4

Apport

1.Introduction :

La classification est une des tâches centrales de l'étape de fouille de données dans le processus d'extraction de connaissances dans les bases de données. Le problème de la classification est traité dans plusieurs communautés de recherche qui se découvrent et s'enrichissent mutuellement : statistiques, reconnaissances de formes, apprentissage automatique, réseaux de neurones et raisonnement à partir de cas.

Il existe de nombreux domaines d'application de ce problème : l'attribution de crédit bancaire, la reconnaissance de gènes, la prédiction de sites archéologiques, le diagnostic médical, etc. Plusieurs méthodes de classification supervisée publiées dans la littérature s'appuient sur des techniques différentes : inférence bayésienne, plus proches voisins, arbres de décision, réseaux de neurones ou treillis de concepts. Nous nous intéressons dans cet article aux méthodes basées sur les treillis de concepts.

2.Problématique :

le problème est défini comme suit:

Étant donné un ensemble de données d'apprentissage $D = \{(x_i, Y_i)\ i = 1, \dots, n\}$, l'objectif consiste à induire un classifieur pour effectuer la correspondance :

$\Phi: X \to 2^Y$, où Y est un ensemble fini de classes et X un ensemble d'objet, de manière à optimiser les performances de classification.

Dans la classification multi label hiérarchique, La structure des hiérarchies peut être classée en deux catégories principale, arborescente et graphe dirigé acyclique (DAG) et dans les deux cas chaque noeud de la hiérarchie représente une et une seule classe. Néanmoins, il est possible de représenter une autre forme de structure hiérarchique des classes qui est celle de la hiérarchie

entre l'ensemble des parties 2^Y de l'ensemble des classe Y. Et pour cela, les treillis de Galois semblent être la meilleure solution.

la problématique de la classification multi label hiérarchique est naturellement liée à la notion de concept. En effet, la classification multi label consiste à regrouper les objets ayant les même classes. La notion de concept formel qui est un regroupement maximal d'objets possédant des propriétés en commun est par conséquent très proche, et le treillis de concepts peut être utilisé dans des applications de classification multi label.

3. Notre approche de classification a base d'ACF:

La classification basée sur l'analyse de concepts formels consiste à proposer des modèles appelés classifieurs à partir des données permettant de prédire les classes des futures données. Elle vise à regrouper tous les groupements possibles entre les classes.

Comme c'est le cas dans la classification supervisée, cette opération est réalisée en deux parties : une phase d'apprentissage dans laquelle un classifieur est construit pour décrire un ensemble prédéterminé de classes à partir d'un ensemble d'apprentissage et une phase de classification où le classifieur construit est utilisé pour associer un ensemble de classes à chaque nouvel objet.

4 .Rappel du processus de classification :

D'une manière générale, la classification par apprentissage pourrait se décomposer en quatre principales étapes :

- **La 1ère :** faire classer un échantillon d'individus par un expert. Cet échantillon est Désigné par le nom de base d'apprentissage.
- **La 2ème :** concevoir et mettre en œuvre un algorithme appelé classifieur qui parvient à reproduire la classification de l'échantillon d'apprentissage.
- **La 3ème :** Evaluer la qualité du classifieur en l'appliquant à un ensemble d'individus classés par l'expert mais qui n'ont pas été utilisés au cours de la phase d'apprentissage.
- **La 4ème :** Si le test est satisfaisant, appliquer la méthode de la 2ème étape à l'ensemble des objets à classer.

5 .Apprentissage supervisé :

L'apprentissage est dit supervisé lorsque les données utilisées dans le processus sont déjà catégorisées, et que les algorithmes d'apprentissage automatiques doivent s'en servir pour prédire un résultat en vue de pouvoir le faire plus tard lorsque les données ne seront plus catégorisées

Le processus se déroule en deux étapes : la phase d'apprentissage et la phase de classement. Avant la phase d'apprentissage, un expert doit, au préalable selon un objectif bien défini, étiqueter les exemples soigneusement sélectionnés. Puis, un modèle data Mining est construit à partir de ces données en utilisant des algorithmes d'apprentissage automatiques. La phase de classement consiste à tester les performances du modèle à prédire correctement les étiquettes de nouvelles instances

5.1 .Phase d'apprentissage :

La phase d'apprentissage consiste à construire un modèle (ou classifieur) $\hat{f}$, qui approxime au mieux la fonction f à partir d'un ensemble d'exemples sélectionnés de manière aléatoire dans l'ensemble d'apprentissage.

5.2 .Phase de classement :

Dans la phase de classement le modèle appris $\hat{f}$ est utilisé pour affecter une classe à chaque nouvel exemple O_i. Etant donné O_k, il s'agit de déterminer $\widehat{C_k} = \hat{f}\ (O_k)$. Pour valider le modèle appris, un ensemble test contenant des exemples dont on connait la classe C_i est utilisé.

Le modèle commet une erreur lorsque $\widehat{C_k}$ est différent de $C_k = f(O_k)$, le modèle appris $\hat{f}$ approxime au mieux f lorsque le nombre d'erreurs calculé tend vers 0. Ce qui revient à déterminer le taux d'erreurs du modèle qui exprime le pourcentage d'instances mal classées par rapport au nombre total d'exemples.

Si le modèle est validé sur l'ensemble test, alors il est utilisé pour classer les exemples dont on connait pas la classe.

6. Phases de classification basée sur l'analyse de concepts formels :

6.1. Phase d'apprentissage :

On commence notre phase d'apprentissage par l'organisation des données d'apprentissage sous forme d'un contexte formel reliant les objets aux classes. Cette relation est représentée sous

forme d'une table. Dans notre cas, les lignes représentent les objets et les colonnes les classes, chaque cellule exprime une valeur appartenant a {0,1}.

La deuxième phase consiste à construire le treillis de Galois correspondant pour mettre en évidence les relations qui existent entre les classes, la construction du treillis de concepts formels d'une relation binaire donnée peut être décomposée en trois parties **[1]** :

1. L'énumération des rectangles maximaux (les concepts formels),

2. La recherche de la relation d'ordre partiel entre ces concepts formels,

3. La construction du diagramme de HASSE correspondant au treillis.

Dans ce qui suit le treillis de Gallois correspondant au contexte formel présenté dans la Table

classes / Instances	Prédateur	Vole	Ovipare	Mammifère
Lion	×			×
Rouge-gorge		×	×	
Aigle	×	×	×	
Autruche			×	
Lièvre				×

Table 12: Contexte formel

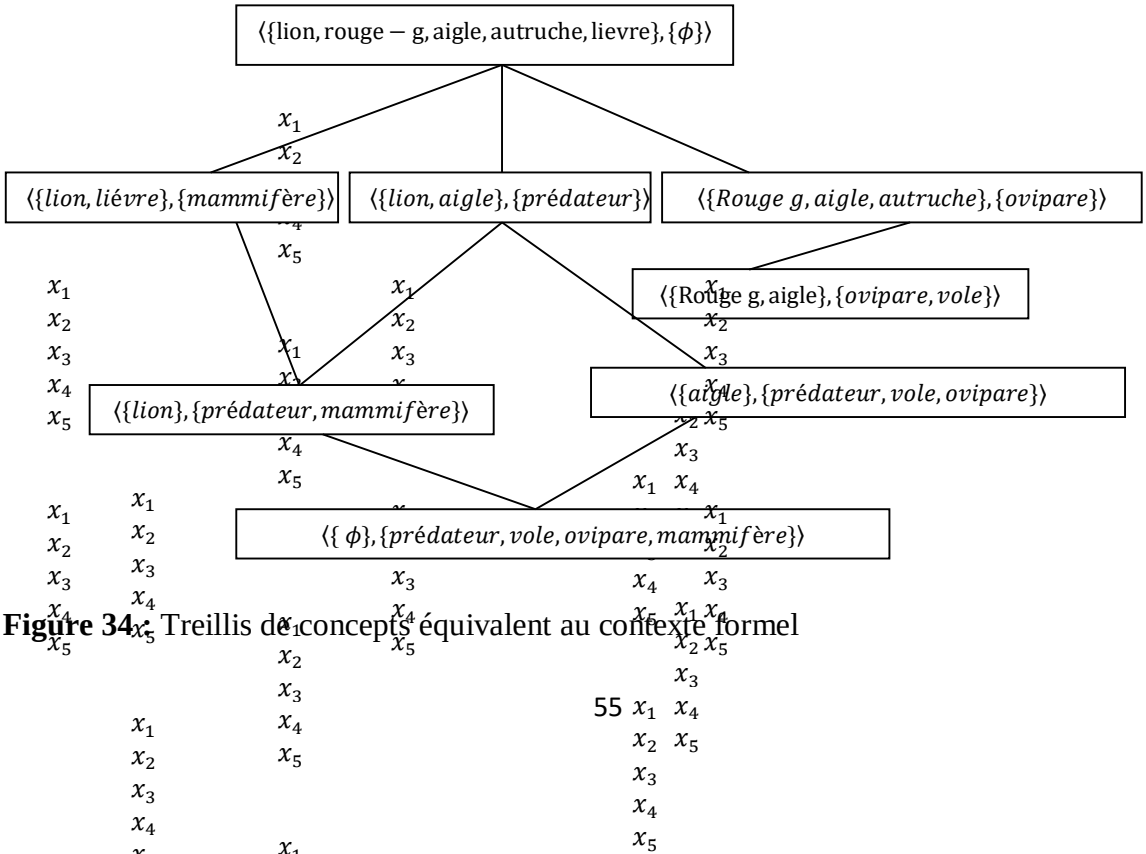


Figure 34: Treillis de concepts équivalent au contexte formel

Ainsi on obtient une hiérarchie entre les concepts formels. Entre ces concepts formels, il y a une relation d'ordre partiel c.à.d. la relation "*Sous-concept*, *Super-concept*".

Etant donné deux concepts formels $\langle X, A\rangle$ et $\langle Y, B\rangle$. On dit que $\langle X, A\rangle$ est un *sous-concept* de $\langle Y, B\rangle$, (dualement $\langle Y, B\rangle$. est un *super-concept* de $\langle X, A\rangle$) si l'extension de $\langle X, A\rangle$ est un *sous ensemble* de l'extension de $\langle Y, B\rangle$. c.à.d. $X \subseteq Y$(dualement : l'intension de $\langle X, A\rangle$ est un *sur-ensemble* de l'intension de $\langle Y, B\rangle$. c.à.d. $A \supseteq B$).

Reprenons l'exemple précédent. Etant donné les deux concepts formels suivants:

⟨{Aigle},{prédateur, vole, ovipare}⟩ et ⟨{Aigle, Rouge-gorge},{vole, ovipare}⟩. ⟨{Aigle},{prédateur, vole, ovipare}⟩ est un sous concept de ⟨{Aigle, Rouge-gorge},{vole, ovipare}⟩. Dualement, ⟨{Aigle, Rouge-gorge},{vole, ovipare}⟩ est un super-concept de $\langle\{\text{Aigle}\}, \{\text{predateur}, \text{vole}, \text{ovipare}\}\rangle$. L'extension {Aigle} du sous-concept est un sous ensemble de l'extension {Aigle, Rouge-gorge} du super-concept. De la même manière l'intension {prédateur, vole, ovipare} du sous-concept est un sur-ensemble de l'intension {vole, ovipare} du super-concept.

Pour chaque noeud de la hiérarchie (treillis), un ou plusieurs classifieurs mono label sont induit pour prédire si une instance appartient ou pas à une classe présente dans l'intension du concept formel (noeud)).
Les classifieurs sont induits en suivant les contraintes suivantes:

1. Un classifieur mono label h est induit pour chaque classe contenue dans l'intension Y du concept bottom (le plus générale), et le résultat de la prédiction sera la combinaison de tous les classifieurs mono label induits:
   Soit n le nombre de classes dans l'intension Y du bottom, $h_i(x)$ le classifieur monolabel associé à la classe i, tel que :

$$h_i(x) = \begin{cases} 0 & Si\ x\ n'appartient\ \ pas\ \text{à}\ la\ classe\ i \\ 1 & Sinon \end{cases}$$

   alors:

   $H_Y(x) = \bigwedge_{i=0,n} h_i(x)$ tel que $H_Y(x)$ est la prédiction que Y satisfait l'objet x

$$H_Y(x) = \begin{cases} 0 & x\ n'appartient\ pas\ \text{à}\ l'intension\ Y \\ 1 & Sinon \end{cases}$$

2. Soit C_i un concept formel (un noeud du treillis) et Y_i son intension (Y_i est un ensemble de classes):

- Si C_i contient un seul super concept C_j dans la hiérarchie, alors un classifieur mono label h est induit pour chaque nouvelle classe qui apparait dans Y_i, et le résultat de la prédiction sera la combinaison de ces classifieurs mono label avec le classifieur du super concept C_j :

  Soit n le nombre de nouvelles classes apparue dans l'intension Y_i :

$$H_{Y_i}(x) = H_{Y_j}(x) \bigwedge_{i=0,n} h_i(x)$$

  - $H_{Y_i}(x)$: La prédiction pour l'intension Y_i du sous concept C_i

$$H_{Y_i}(x) = \begin{cases} 0 & x\ n'appartient\ pas\ \text{à}\ l'intension\ Y_i \\ 1 & Sinon \end{cases}$$

  - $H_{Y_j}(x)$: La prédiction pour l'intension Y_j du super concept C_j

$$H_{Y_j}(x) = \begin{cases} 0 & x\ n'appartient\ pas\ \text{à}\ l'intension\ Y_j \\ 1 & Sinon \end{cases}$$

  - $h_i(x)$: La prédiction pour chaque classe apparue (qui appartient à Y_i mais pas à Y_j)

$$H_i(x) = \begin{cases} 0 & x\ n'appartient\ pas\ \text{à}\ la\ classe\ \ i \\ 1 & Sinon \end{cases}$$

Exemple : en reprenant l'exemple précédent:

- Les noeuds $\langle\{mammifère\}\rangle$, $\langle\{prédateur\}\rangle$ et $\langle\{ovipare\}\rangle$ n'ont qu'un seul super concept et dans chacun de ces noeuds les classes sont nouvelles, donc il faut induire un classifieur mono label pour chacune de ces classes.
- Idem pour le noeud $\langle\{ovipare, vole\}\rangle$ qui a aussi un seul super concept dont l'intension est $\langle\{ovipare\}\rangle$ alors on va construire un classifieur mono label pour chaque nouvelle classe apparue. Et dans ce cas la seule classe apparue est $\langle vole\rangle$, et le résultat de la prédiction sera donc :

$$H_{\langle\{ovipare,vole\}\rangle}(x) = H_{\langle vole\rangle}(x) \bigwedge H_{\langle Ovipare\rangle}(x)$$

3. Soit C_i un concept formel (un noeud du treillis) et Y_i son intension (Y_i est un ensemble de classes):

- Si C_i contient plusieurs super concept C_j $(j = 0, \ldots, n)$ dans la hiérarchie, alors la prédiction sera la combinaison de ces classifieurs des supers concepts C_j :

$$H_{Y_i}(x) = \bigwedge_{j=0,n} H_{Y_j}(x)$$

Exemple:

le noeud $\langle\{prédateur, vole, ovipare\}\rangle$ a deux supers concepts, $\langle\{ovipare, vole\}\rangle$ et $\langle\{prédateur\}\rangle$, le résultat de prédiction pour noeud sera la combinaison des prédiction de ces supers concepts:

$$H_{Y_i}(x) = \bigwedge_{j=0,n} H_{Y_j}(x) = H_{\langle\{ovipare,vole\}\rangle}(x) \bigwedge H_{\langle\{prédateur\}\rangle}(x)$$

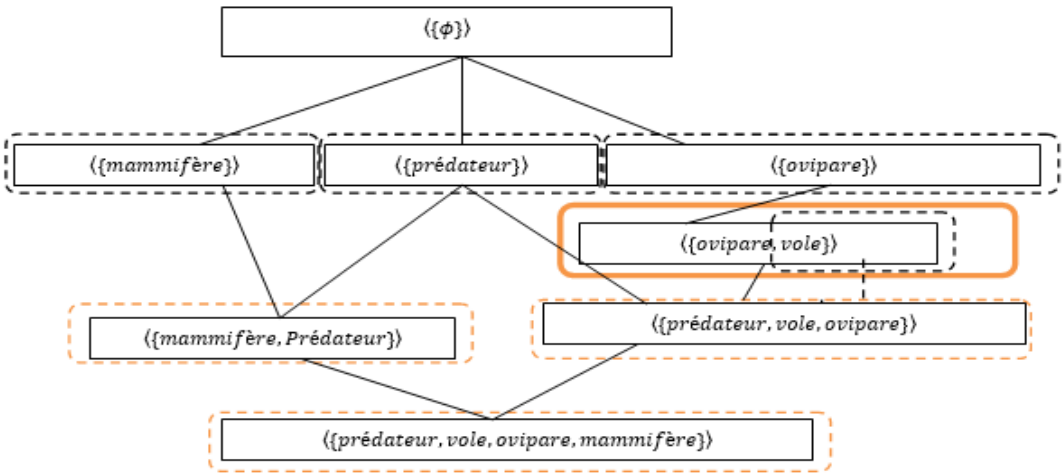


Figure 35 : les classifieurs induit pour l'exemple précédent

- - - - classifieur pour une nouvelle classe

classifieur qui combine la prédiction d'une nouvelle classe avec la prédiction du super concept

classifieur qui combine les prédictions des classifieurs des super concepts

6.2. Phase de classification:

Une instance est classifiée de manière descendante (TOP-DOWN).

- Si l'intension du TOP est un ensemble vide alors on passe directement au sous concept.
- Sinon, l'instance est soumise aux classifieurs monolabel induit pour chaque classe réspectivement, et le résultat de la prédiction sera la combinaison des différentes sortie des ces classifieurs (comme vu dans la phase apprentissage).

Lorsqu'elle est affectée, l'instance sera soumise à une nouvelle classification qui concerne chaque intension de chaque sous concept du concept TOP (nœud racine). Dans notre exemple, l'instance sera soumise à trois classifieurs mono label afin de prédire si elle appartient aux classes $mammifère$, $prédateur$ et $ovipare$ respectivement. Cette procédure est répétée jusqu'à ce qu'on atteint le noeud bottom ou jusqu'à ce qu'aucune prédiction n'est possible.

Supposons qu'une instance inconnue est prédite dans les classes *mammifère* et *prédateur* mais pas dans la classe *ovipare*. Ainsi cette instance n'a pas besoin de continuer le processus de classification de niveau inférieur concernant tous les sous concept du concept contenant *ovipare* comme intension $\langle\{ovipare, vole\}\rangle, \langle\{prédateur, vole, ovipare\}\rangle,$ $\langle\{prédateur, vole, ovipare, mammifère\}\rangle$

En prenant en considération la classification multi label une instance inconnue peut être prédite par plusieurs classifieurs mono label, alors l'instance sera automatiquement prédite par le classifieur du sous concept (reliant ces deux concepts). Puisque l'instance est prédite dans les classes *mammifère* et *prédateur* respectivement alors elle est aussi prédite dans l'intension du sous concepts les reliant $\langle\{mammifère, Prédateur\}\rangle$.

7 .Critères de comparaison des méthodes de classification :

Nous allons caractériser les méthodes de classification basées sur les treillis de concepts en fonction des critères suivants : principe de génération des hypothèses apprises, validation du modèle, complexité théorique.

Le principe de génération détermine l'espace de recherche considéré pour engendrer les hypothèses. La manière par laquelle les hypothèses apprises sont obtenues et évaluées est décrite. Le critère d'arrêt permettant de limiter l'exploration de l'espace de recherche est présenté. Dans les cas qui nous intéressent, l'apprentissage est dirigé par les données avec deux approches (duales) possibles : l'approche descendante ou par spécialisations successives qui consiste à détecter à chaque étape les sous-ensembles spécifiques maximaux d'un ensemble donné d'objets; et l'approche ascendante ou par généralisations successives qui consiste à construire à chaque étape les propriétés communes les plus spécifiques à un sous-ensemble d'objets.

- Le critère de validation est basé sur le calcul du taux de précision qui mesure la capacité du modèle appris à prédire correctement la classe des exemples de l'ensemble test. Ce taux est égal au pourcentage des exemples de l'ensemble test qui sont correctement classés par le modèle.
- Le troisième critère de comparaison consiste à calculer un ordre de complexité en temps et en espace, en fonction de la taille des données et des paramètres de classification. Le critère de complexité permet de juger de l'efficacité des systèmes de classification. Lorsque les performances de précision et de compréhensibilité entre systèmes de classification sont égales, le critère de complexité en temps et en espace peut permettre de préférer le système ayant la plus petite complexité.

D'autres critères, tels que le langage de description ou le langage de représentation des hypothèses apprises, peuvent aussi être utilisés. Ils permettent de juger la compréhension par l'utilisateur des connaissances apprises et des décisions produites par le système.

Conclusion :

Dans la classification multi-label, chaque instance peut appartenir à une ou plusieurs classes simultanément. Très souvent dans le cas d'application réelle, les classes ne sont pas indépendantes les unes des autres. Nous avons proposé, dans le cadre de ce mémoire, une approche basée sur l'analyse de concepts formels qui nous permet, à la fois, de représenter les classes sous forme d'une hiérarchie (matérialisée par un treillis de Galois) et de visualiser les différentes relations entre les classes.

L'utilisation du treillis de Galois permet de décomposer le concept à caractériser en sous-concepts. Il est alors plus aisé de rechercher une bonne caractérisation d'un ensemble réduit de données. Dans cette approche, les nœuds du treillis qui caractérisent chacun un sous-concept, possèdent une double représentation. L'une est constituée par l'hypothèse qui généralise les exemples du sous-concept concerné (l'intension). L'autre utilise l'ensemble des exemples du sous-concept (l'extension).

Conclusion générale

Au cours de ce mémoire, Nous avons proposé une approche de classification basée sur l'analyse de concepts formels qui permet d'obtenir une hiérarchie entre les concepts formels. La classification basée sur l'analyse de concepts formels consiste à construire des modèles appelés classifieurs à partir des données permettant de prédire les classes des futures données. Elle vise à regrouper tous les groupements possibles entre les classes.

Après avoir présenté la classification multi label nous avons constaté que les algorithmes proposés dans ce domaine ne tiennent pas compte des relations existantes entres les différentes classes et n'exploitent pas leurs structures hiérarchiques.

La classification multi-label hiérarchique traite des problèmes où les classes sont organisées sous forme d'une hiérarchie cette structure nous a fait penser à l'analyse de concepts formels et aux treillis de Galois.

L'utilisation du treillis de Galois permet d'avoir une représentation de la réalité étudiée (contexte formel). Cette structure en treillis met en évidence les relations qui existent entre les objets d'une part mais aussi les relations entre les propriétés décrivant ces objets d'une autre part. Il est alors plus aisé de rechercher les propriétés demandées par l'utilisateur.
Ce projet a fait objet d'une expérience intéressante qui nous a permis de renforcer et d'enrichir nos connaissances concernant l'intelligence artificielle.

Références bibliographiques et webographies :

[1] : Violaine Godin, Formation d'auxiliaire de Bibliothèque « LA CLASSIFICATION DECIMALE DE DEWEY » 08/01/2015.

[2] : dspace.univ-tlemcen.dz/bitstream/112/1045/4/Memoire.pdf KoudriMohamme – 2011.

[3] : Pascale Kuntz, Frank Meyer, Sélection d'une méthode de classification multi-label pour un système interactif, HAL Id : hal-00908610

[4] : Zhu X. Semi-supervised learning literature survey (Tech. Rep. 1530). Computer Sciences, University of Wisconsin – Madison, 2007.

[5] : YasmineHanane et ZegganeMokhtar ,Algorithme d'apprentissage pour la classification de document 2009, {romain.guigoures, marc.boulle}@orange-ftgroup.com

[6] : Nasierding G, Kouzani A. Image to text translation by multi-label classification. In: Advanced Intelligent Computing Theories and Applications with Aspects of Artificial Intelligence, Lecture Notes in Computer Science, vol. 6216, Berlin/Heidelberg: Springer; 2010, 247–254

[7] : Kumar N, Berg AC, Belhumeur PN, Nayar SK. Attribute and simile classifiers for face verification. In: IEEE International Conference on Computer Vision (ICCV), 2009

[8] : Wang J, Zhao Y, Wu X, Hua XS. A transductive multi-label learning approach for video concept detection. Pattern Recogn 2010, 44:2274–2286

[9] : Trohidis K, Tsoumakas G, Kalliris G, Vlahavas I. Multi-label classification of music into emotions. In: Proceedings of the 9th Intern. Conf. on Music Information Retrieval (ISMIR 2008); 2008, 325–330.

[10] : Pachet F, Roy P. Improving multilabel analysis of music titles : a large-scale validation of the correction approach. IEEE Trans Audio Speech Lang Proc 2009, 17:335–343.

[11] : Sobol-Shikler T, Robinson P. Classification of complex information : inference of co-occurring affective states from their expressions in speech. IEEE Trans Pattern Anal Mach Intell 2010, 32 :1284–1297.

[12] : Barutcuoglu Z, Schapire RE, Troyanskaya OG. Hierarchical multi-label prediction of gene function. Bioinformatics 2006, 22 :830–836.

[13] : Chan A, Freitas AA. A new ant colony algorithm for multi-label classification with applications in bioinfomatics. In: GECCO'06: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, New York, USA; 2006, 27–34.

[14] : Duwairi R, Kassawneh A. A framework for predicting proteins 3D structures. In: IEEE/ACS International Conference on Computer Systems and Applications (AICCSA'08), Washington, DC, USA; 2008, 37–44.

[15] : Mammadov MA, Rubinov AM, Yearwood J. The study of drug-reaction relationships using global optimization techniques. Optim Method Softw 2007, 22 :99–126

[16] : Ukwatta E, Samarabandu J. Vision based metal spectral analysis using multi-label classification. In : Canadian Conference on Computer and Robot Vision (CRV '09); 2009, 132–139.

[17] : Tang L, Liu H. Scalable learning of collective behavior based on sparse social dimensions. In : Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM'09), New York, NY, USA ; 2009, 1107–1116.

[18] : Peters S, Denoyer L, Gallinari P. Iterative annotation of multi-relational social networks. In : International Conference on Advances in Social Networks Analysis and Mining (ASONAM); 2010, 96–103.

[19] : López VF, de la Prieta F, Ogihara M, Wong DD. A model for multi-label classification and ranking of learning objects. Expert Syst Appl 2012, 39: 8878–8884.

[20] : Shao H, Li G, Liu G, Wang Y. Symptom selection for multi-label data of inquiry diagnosis in traditional Chinese medicine. Sci China Ser F-Info Sci 2010, 1:1–13.

[21] : Abbas Q, Celebi M, Serrano C, García IF, Ma G. Pattern classification of dermoscopy images : a perceptually uniform model. Pattern Recogn 2013, 46 :86–97.

[22] ASHBURNER, M., BALL, C., BLAKE, J., BOTSTEIN, D., BUTLER, H., CHERRY, M., DAVIS, A., DOLINSKI, K., DWIGHT, S., EPPIG, J., HARRIS, M., HILL, D., ISSEL-TARVER, L., KASARSKIS, A., LEWIS, S., MATESE, J., RICHARDSON, J., RINGWALD, M., RUBIN, G., AND SHERLOCK, G. 2000.

Gene ontology: tool for the unification of biology. Nature Genetics 25, 1, 25-29.

[23] MCCALLUM, A., ROSENFELD, R., MITCHELL, T. M., AND NG, A. Y. 1998.

Improving text classification by shrinkage in a hierarchy of classes. In Proceedings of the Fifteenth International Conference on Machine Learning. Morgan Kaufmann Publishers Inc., 359-367.

[24] DUMAIS, S. AND CHEN, H. 2000. Hierarchical classification of web content. In Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 256-263.

[25] SARINNAPAKORN, K. AND KUBAT, M. 2007. Combining subclassifiers in text categorization: A dst-based solution and a case study. IEEE Transactions on Knowledge and Data Engineering 19, 12, 1638-1651.

[26] LEWIS, D. D., YANG, Y., ROSE, T. G., AND LI, F. 2004a. Rcv1: A new benchmark collection for text categorization research. Journal of Machine Learning Research 5, 361-397.

[27] HERSH, W., BUCKLEY, C., LEONE, T. J., AND HICKAM, D. 1994.

OHSUMED : an interactive retrieval evaluation and new large test collection for research. In SIGIR '94 : Proceedings of the 17th annual international ACM SIGIR conférence on Research and development in information retrieval. Springer-Verlag New York, Inc., 192-201.

[28] MEWES, H. W., ALBERMANN, K., HEUMANN, K., LIEBL, S., AND PFEIFFER, F. 1997. Mips: a database for protein sequences, homology data and yeast genome information. Nucleic Acids Res 25, 1, 28-30.

[29] HARRIS, C. J. 2004. The gene ontology (go) database and informatics resource gene ontology consortium 32 (supplement 1): 258 - nucleic acids research.Nucleic Acids Res. 1, 32, D258-D261.

.

[30] STENGER, B., THAYANANTHAN, A., TORR, P. H. S., AND CIPOLLA, R.

2007. Estimating 3d hand pose using hierarchical multi-label classification. Image Vision Comput. 25, 12, 1885-1894.

[31] JENSEN, L. J., GUPTA, R., BLOM, N., DEVOS, D., TAMAMES, J., KESMIR, C., NIELSEN, H., STAERFELDT, H. H., RAPACKI, K., WORKMAN, C., AN-

DERSEN, C. A., KNUDSEN, S., KROGH, A., VALENCIA, A., AND BRUNAK,

S. 2002. Prediction of human protein function from post-translational modifications and localization features. Journal of Molecular Biology (JMB) 319, 5, 1257-1265.

[32] RILEY, M. 1993. Functions of the gene products of escherichia coli. Microbiol. Mol. Biol. Rev. 57, 4, 862-952.

[33] WEINERT, W. R. AND LOPES, H. S. 2004. Neural networks for protein classification. Applied Bioinformatics 3, 1, 41-48.

[34] BERMAN, H., WESTBROOK, J., FENG, Z., GILLILAND, G., BHAT, T. N., WEISSIG, H., SHINDYALOV, I., AND BOURNE, P. 2000. The protein data bank. Nucleic Acids Research 28, 1, 235-242.

[35] SUN, A. AND LIM, E.-P. 2001. Hierarchical text classification and evaluation. In Proceedings of the 2001 IEEE International Conference on Data Mining. ICDM '01. IEEE Computer Society, Washington, DC, USA, 521-528.

[36] KOLLER, D. AND SAHAMI, M. 1997. Hierarchically classifying documents using very few words. In Proceedings of ICML-97, 14th International Conference on Machine Learning, D. Fisher, Ed. Morgan Kaufmann Publishers, San Francisco, US, 170-178.

[37] Trohidis K, Tsoumakas G, Kalliris G, Vlahavas I. Multi-label classification of music into emotions. In: Proceedings of the 9th Intern. Conf. on Music Information Retrieval (ISMIR 2008); 2008, 325–330.

[38] COSTA, E. P., LORENA, A. C., CARVALHO, AND FREITAS, A. A. 2007. A review of performance evaluation measures for hierarchical classifiers. In 2007 AAAI Workshop, Vancouver. AAAI Press.

[39] SUN, A., LIM, E.-P., AND LIU, Y. 2009. On strategies for imbalanced text classification using svm: A comparative study. Decision Support Systems 48, 1, 191-201.

[40] SILLA, C. AND FREITAS, A. 2010. A survey of hierarchical classification across different application domains. Data Mining and Knowledge Discovery .

[41] EISNER, R., POULIN, B., SZAFRON, D., LU, P., AND GREINER, R. 2005. Improving protein function prediction using the hierarchical structure of the gene ontology. In Proceedings of IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB).

[42] FAGNI, T. AND SEBASTIANI, F. 2007. On the selection of negative examples for hierarchical text categorization. In Proceedings of the 3rd language technology conference. 24-28.

[43] KIRITCHENKO, S., MATWIN, S., AND FAMILI, A. F. 2005. Functional annota-tion of genes using hierarchical text categorization. In in Proceedings of the BioLINK SIG: Linking Literature, Information and Knowledge for Biology (held at ISMB-05).

[44] B. Ganter and R. Wille. Formal Concept Analysis. Springer, mathematical foundations edition, 1999.

[45] R**.** Wille. Restructuring lattice theory : an approach based on hierarchies of ordered sets, pages 445_470, 1982

[46] Karl Erich Wolff. A first course in formal concept analysis how to understand line diagrams. 2003.

[47] Wille, R. (1982). Restructuring lattice theory : an approach based on hierarchies of concepts. Ordered sets, pages 445–470.

[48] Bernhard Ganter et Rudolf Wille. Formal concept analysis mathematical foundations. Springer, 1999. ISBN 978-3-540-62771-5.

[49] Ivo Duntsch et Gunther Gediga. Approximation operators in qualitative data analysis. In Harrie de Swart, Ewa Or lowska, Gunther Schmidt, et Marc Roubens, editors, Theory and Applications of Relational Structures as Knowledge Instruments : COST Action 274, TARSKI. Revised Papers, pages 214{230. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003. ISBN 978-3-540-24615-2. doi :10.1007/978-3-540-24615-2fn g10.

[50] Alain Guffenoche et Iven Van Mechelen. Galois approach to the induction of concepts.In I. Van Mechelen, J. Hampton, R. Michalski, et P. Theuns, editors, Categories and Concepts : Theoretical Views and Inductive Data Analysis. Academic Press,1993.

[51] Michel Chein. Algorithme de recherche des sous-matrices premières d'une matrice. Bull. Math. Soc. Sc. Math. de Roumanie, 1(13) :21{25, 1969

[52] E. M. Norris. An algorithm for computing the maximal rectangles in a binary relation. Revue Roumaine de Mathématiques Pures et Appliquées, 23(2) :243{250,1978.

[53] Bernhard Ganter. Two basic algorithms in concept analysis. FB4{Preprint 831, THDarmstadt, 1984

[54] Bernhard Ganter et Klaus Reuter. Finding all closed sets : A general approach.Order, 8(3) :283{290, 1991.

[55] J. P. Bordat. Calcul pratique du treillis de galois d'une correspondance. Mathématiques et Sciences Humaines, 96 :31{47, 1986.

[56] Sergei O Kuznetsov. A fast algorithm for computing all intersections of objects in a finite semi-lattice. Automatic Documentation and Mathematical Linguistics, 27 (5) :11{21, 1993.

[57] Robert Godin, Rokia Missaoui, et Hassan Alaoui. Incremental concept formation algorithms based on galois (concept) lattices. Computationnel Intelligence, 11(2) :246{267, 1995. ISSN 1467-8640.

[58] Claudio Carpineto et Giovanni Romano. A lattice conceptual clustering system and its application to browsing retrieval. Machine Learning, 24(2) :95{122, 1996.

[59] Lhouari Nourine et Olivier Raynaud. A fast algorithm for building lattices. Information Processing Letters, 71(5-6) :199{204, September 1999.

[60] G. Stumme, R. Taouil, Y. Bastide, N. Pasquier, et L. Lakhal. Fast computation of concept lattices using data mining techniques. In M. Bouzeghoub, M. Klusch,W. Nutt, et U. Sattler, editors, Proc. 7th Intl. Workshop on Knowledge Representation Meets Databases, 2000.

[61] Gerd Stumme, Rafik Taouil, Yves Bastide, Nicolas Pasquier, et L. Lakhal. Computing iceberg concept lattices with titanic. Data & Knowledge Engineering, 42(2) :189 { 222, 2002

[62] Petko Valtchev et Rokia Missaoui. Building Concept (Galois) Lattices from Parts : Generalizing the Incremental Methods, pages 290-303. Springer Berlin Heidelberg,Berlin, Heidelberg, 2001.

[63] P. Valtchev, R. Missaoui, et P. Lebrun. A partition-based approach towards constructing galois (concept) lattices. Discrete Mathematics, 256(3) :801 {829, 2002.

[64] Dean van der Merwe, Sergei Obiedkov, et Derrick Kourie. AddIntent : A New Incremental Algorithm for Constructing Concept Lattices, pages 372{385. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[65] L.A Zadeh. Fuzzy sets as a basis for a theory of possibility. Fuzzy Sets and Systems,1(1):3 { 28, 1978. ISSN 0165-0114.

[66] Didier Dubois, Florence Dupin de Saint Cyr Bannay, et Henri Prade. A possibility theoretic view of formal concept analysis. Fundamenta Informaticae,75(14) :195{213, 2007.

[67] [Guenoche, 1990] Guenoche, A. (1990). Construction du treillis de galois d'une relation binaire.Mathmatiques, Informatique et Sciences Humaines, 28ème année(109) :41–53.

[68] [Guenoche and Mechelen, 1993] Guenoche, A. and Mechelen, I. V. (1993). Galois approach to the induction of concepts. In Mechelen, I. V., Hampton, J., Michalski, R., and Theuns, P.,editors, Categories and Concepts. Theoretical Views and Inductive Data Analysis, pages 287–308. Academic Press, London

[69] : [Kuznetsov, 1993] Kuznetsov, S. (1993). A fast algorithm for computing all intersections of objects in a finite semi-lattice. Automatic Documentation and Mathematical Linguistics, 27(5) :11–21.

[70] : Dowling, 1993] Dowling, C. E. (1993). On the irredundant generation of knowledge spaces. Journal of Mathematical Psychology, 37(1) :49–62.

[71] : [Nguifo and Njiwoua, 1998] Nguifo, E. M. and Njiwoua, P. (1998). Using Lattice-based

Feamework as a Tool for Feature Extraction, chapter 13, pages 205–216. Kluwer Academic Publishers.

[72] Lindig, C. (2000). Fast concept analysis. In Stumme, G., editor, Working with Conceptual Structures - Contributions to ICCS 2000, Aachen, Germany.

[73] : Martin, B. and Eklund, P. W. (2008). From concepts to concept lattice : A border algorithm for making covers explicit. In [Medina and Obiedkov, 2008], pages 78–89.

[74] : Szathmary, L., Valtchev, P., Napoli, A., and Godin, R. (2008).Constructing iceberg lattices from frequent closures using generators. In Boulicaut, J.-F.Berthold, M. R., and Horváth, T., editors, Discovery Science, 11th International Conference,DS 2008, Budapest, Hungary, October 13-16, 2008. Proceedings, volume 5255 of Lecture Notes in Computer Science, pages 136–147. Springer.

[75] : Ganter, B. and Reuter, K. (1991). Finding all closed sets : A general approach. Order, 8(3) :283–290.

[76] : Godin, R., Missaoui, R., and April, A. (1993). Experimental comparison of navigation in a galois lattice with conventional information retrieval methods. International Journal of Man-machine Studies, 38 :747–767.

[77] Carpineto, C. and Romano, G. (1996). A lattice conceptual clustering system and its application to browsing retrieval. Machine Learning, 24(2) :95–122.

[78] Carpineto, C. and Romano, G. (2004a). Concept Data Analysis : Theory and Applications. John Wiley & Sons.

[79] Kuznetsov, S. O. and Obiedkov, S. A. (2002). Comparing Performance of Algorithms for Generating Concept Lattices. Journal of Experimental & Theoretical Artificial Intelligence, 14 :189–216.

[80] Stumme, G., Taouil, R., Bastide, Y., and Lakhal, L. (2001). ConceptualClustering with Iceberg Concept Lattices. In Proceeding GI-Fachgruppentreffen Maschinelles Lernen (FGML'01), Universitat Dortmund 763.

[81] : Carpineto, C. and Romano, G. (1993). Galois : An ordertheoretic approach to conceptual clustering. Proceedings of 10th International Conference on Machine Learning, Amherst, pages 33–40.

[82] Carpineto, C. and Romano, G. (1996). A lattice conceptual clustering system and its application to browsing retrieval. Machine Learning, 24(2) :95–122.

[83] Godin, R., Mineau, G., and Missaoui, R. (1995a). Incremental structuring of knowledge bases. In Ellis, G., Levinson, R. A., Fall, A., and Dahl, V., editors, Proceedings of the 1st International Symposium on Knowledge Retrieval, Use, and Storage for Efficiency (KRUSE'95), Santa Cruz (CA), USA, pages 179–193. Department of Computer Science, University of California at Santa Cruz.

[84] Godin, R., Mineau, G. W., and Missaoui, R. (1995b). Méthodes de classification conceptuelle basées sur les treillis de Galois et applications. Revue d'intelligence artificielle, 9(2) :105–137.

[85] G. Birkhoff. Lattice Theory, volume 25 of ASM Colloquium Publications. AMS, Providence,RI, 3rd edition, 1967. 1st ed., 1940 ; 2nd ed., 1948 [Godin et al., 1995c] Godin, R., Missaoui, R., and Alaoui, H. (1995c). Incremental Concept Formation Algorithms Based on Galois (Concept) Lattices. Computational Intelligence, 11 :246–267.

Résumé :

L'apprentissage multi-label est un problème d'apprentissage supervisé où chaque instance peut être associée à plusieurs labels cibles simultanément.

La plupart des problèmes de classification traités considèrent que chaque exemple n'est associé qu'à une seule étiquette

La classification multi-labels hiérarchique traite des problèmes où les classes sont organisées sous forme d'une hiérarchie, cette organisation hiérarchique est essentielle parce qu'elle permet à l'utilisateur de se concentrer sur le niveau approprié de détails. A partir de là, nous pouvons légitimement penser à l'analyse de concepts formels et aux treillis de Galois. La classification basée sur l'analyse de concepts formels consiste à construire des modèles appelés classifieur à partir des données permettant de prédire les classes des futures données .

Mots-clés : ACF, classification multi label, classification multi label hiérarchique.

Abstract :

Multi-label learning is a supervised learning problem where each instance can be associated with several target labels simultaneously.

Most of the classification issues treated consider that each example is associated with only one label.

Hierarchical multi-label classification addresses issues where classes are organized as a hierarchy, this hierarchical organization is essential because it allows the user to focus on the appropriate level of detail. From there, we can legitimately think of the analysis of formal concepts and the Galois lattices. Classification based on the analysis of formal concepts consists in building models called classifier from the data making it possible to predict the classes of future data.

Keywords: ACF, multi label classification, hierarchical multi label classification.