Chapitre 3

Classification

3.1 Concepts de base

3.1.1 Définition

La classification est une tâche très importante dans le data mining, et qui consomme beaucoup de recherches pour son optimisation. La classification supervisée est l'une des techniques les plus utilisées dans l'analyse des bases de données. Elle permet d'apprendre des modèles de décision qui permettent de prédire le comportement des exemples futurs.

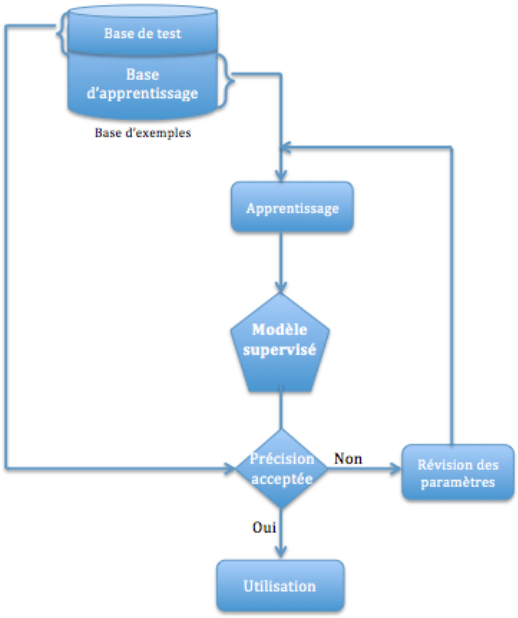
La classification supervisée consiste à inférer à partir d'un échantillon d'exemples classés une procédure de classification. Un système d'apprentissage effectue la recherche d'une telle procédure selon un modèle. Les systèmes d'apprentissage peuvent être basés sur des hypothèses probabilistes (classifieur naïf de Bayes, méthodes paramétriques) ; sur des notions de proximité (plus proches voisins) ; sur des recherches dans des espaces d'hypothèses (arbres de décision, réseaux de neurones).

3.1.2 Organisation

La classification est un processus à deux étapes : une étape d'apprentissage (entrainement) et une étape de classification (utilisation).

Dans l'étape d'apprentissage, un classiffieur (une fonction, un ensemble de règles, ...) est construit en analysant (ou en apprenant de) une base de données d'exemples d'entrainement avec leurs classes respectives. Un exemple $X = (x_1, x_2, .., x_m)$ est représenté par un vecteur d'attributs de dimension m. Chaque exemple est supposé appartenir à une classe prédéfinie représentée dans un attribut particulier de la base de donnée appelé attribut de classe. Puisque la classe de chaque exemple est donnée, cette étape est aussi connue par

l'apprentissage supervisé.



Dans l'étape de classification, le modèle construit dans la première étape est utilisé pour classer les nouvelles données. Mais avant de passer à l'utilisation, le modèle doit être testé pour s'assurer de sa capacité de généralisation sur les données non utilisées dans la phase d'entrainement. Le modèle obtenu peut être testé sur les données d'entrainement elles-mêmes, la précision (le taux de reconnaissance) est généralement élevée mais ne garantit pas automatiquement une bonne précision sur les nouvelles données. En effet, les données d'entrainement peuvent contenir des données bruitées ou erronées (outliers) qui ne représentent pas le cas général et qui tire le modèle vers leurs caractéristiques. Ce cas est appelée le sur-apprentissage ou en anglais *"over fitting"* et qui peut être évité en testant le modèle sur une base de données différente appelée base de test. La base de test est un ensemble d'exemples ayant les mêmes caractéristiques que ceux de la base d'entrainement et qui sont écartés au départ de l'entrainement pour effectuer les tests.
La méthode de prédiction utilisée dépend essentiellement du type d'information prédite c-à-d le type de l'attribut de classe. Si l'attribut est catégoriel ou symbolique (appartient à un ensemble fini), il s'agit de classification. par contre si cet attribut est continu (numérique) il s'agit d'un problème de régression.

Dans le cas de classification, on dispose d'un ensemble X de N données étiquetées représentant un sous ensemble de l'ensemble D de toutes les données possibles. Chaque

donnée x_i est caractérisée par P attributs et par sa classe $y_i \in Y$. Dans un problème de classification, la classe prend sa valeur parmi un ensemble fini. Le problème consiste alors, en s'appuyant sur l'ensemble d'exemples $X = \{(x_i, y_i)/i \in \{1, .., N\}\}$, à prédire la classe de toute nouvelle donnée $x \in D$. On parle de classification binaire quand le nombre de classes $|Y|$ est 2 ; il peut naturellement être quelconque. Dans tous les cas, il s'agit d'un attribut qualitatif pouvant prendre un nombre fini de valeurs. Dans l'absolu, une donnée peut appartenir à plusieurs classes : c'est alors un problème multi-classes. Ici, on considère que chaque donnée appartient à une et une seule classe. Souvent, on utilise le mot "étiquette" comme synonyme de "classe". Un exemple est donc une donnée dont on dispose de sa classe. On utilise donc un ensemble d'exemples classés pour prédire les classes des nouvelles données ; c'est une tâche d'"apprentissage à partir d'exemples", ou "apprentissage supervisé".

3.1.3 Evaluation du modèle

L'apprentissage supervisé utilise une partie des données pour calculer un modèle de décision qui sera généralisé sur l'ensemble du reste de l'espace. Il est très important d'avoir des mesures permettant de qualifier le comportement du modèle appris sur les données non utilisées lors de l'apprentissage. Ces métriques sont calculées soit sur les exemples d'entrainement eux mêmes ou sur des exemples réservés d'avance pour les tests.

La métrique intuitive utilisée est la précision du modèle appelée aussi le taux de reconnaissance. Elle représente le rapport entre le nombre de donnée correctement classées et le nombre total des données testées. L'équation suivante donne la formule utilisée.

$$P = \frac{1}{N} \sum_{i=1}^{N} L(y_i, \hat{f}(x_i))$$

avec :

$$L = \begin{cases} 1 & si \quad y_i = \hat{f}(x_i) \\ 0 & sinon \end{cases}$$

Généralement, la précision est donnée sous forme de pourcentage ce qui nécessite de multiplier la précision de l'équation précédente par 100.

3.1.3.1 Matrice de confusion

La mesure précédente donne le taux d'erreurs commises par le modèle appris ($100 - prcision$) mais ne donne aucune information sur la nature de ces erreurs. Dans la plus part des cas d'application, il est très important de connaître la nature des erreurs commises :

quelle classe est considérée comme quelle classe par le modèle ? Par exemple dans un modèle appris pour des objectifs médicaux, considérer un échantillon non cancéreux alors qu'il l'est, est beaucoup plus grave de considérer un échantillon cancéreux alors qu'il ne l'est pas. Dans le cas de classification binaire, le résultat de test d'un modèle peut être une possibilité parmi quatre :

$$\begin{cases} \hat{f}(x_i) = +1 \quad et \quad y_i = +1 \quad correcte \;\; positive \\ \hat{f}(x_i) = +1 \quad et \quad y_i = -1 \quad fausse \;\; positive \\ \hat{f}(x_i) = -1 \quad et \quad y_i = -1 \quad correcte \;\; négative \\ \hat{f}(x_i) = -1 \quad et \quad y_i = +1 \quad fausse \;\; négative \end{cases}$$

Si le modèle donne une classe positive pour un exemple d'une classe positive, on dit que c'est une classe correcte positive (CP). Si par contre l'exemple appartient à la classe négative on dit que c'est une classe fausse positive (FP). Si le modèle donne une classe négative pour un exemple d'une classe négative, le résultat est une classe correcte négative (CN), si, par contre, la classe de l'exemple est positive le résultat est qualifié de classe fausse négative (FN).

La matrice de confusion est une matrice qui rassemble en lignes les observations (y) et en colonnes les prédictions $\hat{f}(x)$. Les éléments de la matrice représentent le nombre d'exemples correspondants à chaque cas.

TABLE 3.1 – Matrice de confusion pour la classification binaire

Observations (y)	Prédictions ($\hat{f}$)	
	+1	-1
+1	CP	FN
-1	FP	CN

Un modèle sans erreurs aura ses résultats rassemblés sur la diagonale de sa matrice de confusion (CP et CN). Dans le cas multiclasses la matrice sera plus large avec les classes possibles au lieu des deux classes +1 et -1. La précision P du modèle peut être calculée à partir de la matrice de confusion comme suit :

$$P = \frac{CP + CN}{CP + FP + CN + FN}$$

Deux autre mesures sont utilisées dans la littérature : la sensitivité Sv et la spécificité Sp. La sensitivité représente le rapport entre les observations positives correctement prédites et le nombre des observations positives, et la spécificité représente le rapport entre les observations négatives correctement prédites et le nombre total des observations négatives.

$$\begin{cases} Sv = & \frac{CP}{CP+FN} \\ Sp = & \frac{CN}{CN+FP} \end{cases}$$

Une autre métrique calculée à base de la sensitivité et la spécificité est utilisée. C'est la moyenne harmonique.

$$Moyenne\ harmonique = \frac{2 \times Sv \times Sp}{Sv + Sp}$$

3.1.3.2 Évaluation

L'apprentissage d'un modèle de décision se fait à base de plusieurs paramètres, à savoir les exemples d'entrainement et leur nombre, les paramètres de la méthodes utilisée, ...etc. Le choix des valeurs de ces paramètres se fait à travers plusieurs essais et évaluation pour atteindre des performances satisfaisantes du modèle. Les paramètres optimaux pour un modèle donné sont les paramètres qui lui permettent de donner une précision de 100%. Cette situation serait idéale si l'ensemble des exemples représentait parfaitement l'ensemble de tous les exemples possibles. Le modèle appris peut donner une très grande précision face aux exemples d'entrainement, mais se comporte très mal avec les nouveaux exemples. Cela représente un phénomène très connu en apprentissage qui est le sur-apprentissage ou l'apprentissage par coeur. Le sur-apprentissage donne, généralement, des modèles à faible capacité de généralisation, et par conséquent la mesure de précision n'est pas suffisante pour qualifier les performances d'un modèle. Les méthodes d'évaluation permettent de tirer des conclusion sur le comportement d'un modèle face à tout l'espace d'exemples en limitant l'influence des exemples d'entrainement et du bruit qui peut y exister (erreurs d'étiquetage, erreurs d'acquisition, ...) et leur ordre sur le modèle appris.

3.1.3.2.1 Méthode HoldOut La méthode HoldOut suppose que les données d'entrainement sont tout l'espace d'exemples. Elle consiste à diviser l'ensemble des données en deux parties, la première partie est utilisée pour l'entrainement et la deuxième pour les tests. Le test du modèle appris sur la partie de test permet de donner une idée sur son comportement en dehors des exemples d'entrainement et éviter le phénomène de sur-apprentissage. Le modèle qui maximise la précision pour tout l'espace d'exemple est donc celui qui la maximise pour la partie de test du fait que cette partie représente la majorité de l'espace.

Une question importante qui se pose pour cette méthode est comment choisir les deux

parties puisque ce choix a une grande influence sur la qualité du modèle. Le pire est de mettre les exemples positifs dans une partie et les exemples négatifs dans l'autre. La méthode qui suit répond à cette question.

3.1.3.2.2 Validation croisée Pour minimiser l'influence du choix du partitionnement de l'ensemble des exemples, la validation croisée subdivise l'ensemble d'entrainement initial en k sous ensemble disjoints $D_1, D_2, .., D_k$ de même taille. L'entrainement et le test sont effectués k fois. A l'itération i le sous-ensemble D_i est réservé pour le test et le reste des exemples sont utilisés pour entrainer le modèle. La précision finale du modèle est égale à la moyenne des k précisions de test.
La méthode Leave-One-Out est un cas particulier de la validation croisée où $k = N$. Á chaque itération, le modèle est entrainé sur $N - 1$ exemples et testé sur l'exemple exclu de l'entrainement. On obtient à la fin N précisions, la précision du modèle est égale à leur moyenne.

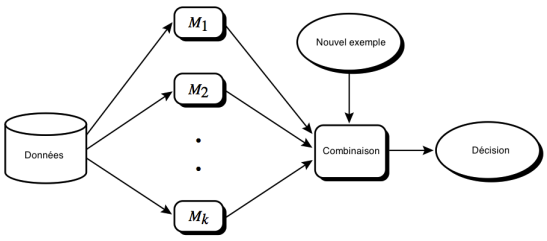
3.1.3.2.3 Le Bootstrap La méthode de Bootstrap, appelée aussi échantillonnage par remplacement, entraine le modèle sur un ensemble de N exemples choisis aléatoirement de l'ensemble des exemples, des exemples peuvent être choisis plus d'une fois et d'autre ne se seront pas choisis du tout. Les exemples non choisis pour l'entrainement sont utilisés pour le test. Cette opération peut être répétée plusieurs fois pour obtenir une précision moyenne du modèle.
Parmi les méthodes de Bootstrap les plus utilisées, la méthode Bootstrap ".632" qui tire son nom du fait que 63.2 % des exemples contribuent à l'entrainement et les restants (36.8%) contribuent aux tests. A chaque prélèvement, un exemple a une probabilité $1/N$ d'être choisi et $(1-1/N)$ de ne pas l'être, et puisqu'on répète le prélèvement N fois, chaque exemple aura une probabilité de $(1-1/N)^N$ de ne pas être choisi du tout dans un ensemble d'entrainement. Si N est grand cette probabilité approche de $e^{-1} = 0.368$. La méthode répète le processus k fois et la précision finale P est donnée par :

$$P = \sum_{i=1}^{k} (0.632 \times Pi_{test} + 0.368 \times Pi_{entr})$$

Où Pi_{test} est la précision du modèle entrainé sur les exemples choisis dans l'itération i, appliqué sur les exemples de test dans la même itération. Pi_{entr} est la précision du même modèle appliqué sur les données d'entrainement.

3.2 Combinaison de modèles

Pour augmenter la précision des modèles obtenus, certaines méthodes combinent plusieurs modèles pour obtenir les décisions.



Deux méthodes sont particulièrement utilisées : Bagging and Boousing.

3.2.1 Bagging

Cette méthode se base sur le Bootstrap. Elle subdivise l'ensemble D d'exemples en n sous-ensembles. À partir de chaque sous-ensemble D_i, on apprend un modèle M_i en utilisant la méthode Bootstrap. L'ensemble de ces modèles forme un modèle composé M_*. Pour classiffier un nouvel exemple, il est exposé à chaque modèle M_i pour obtenir une classe c_{M_i}. Chaque décision est considérée comme un vote. La classe de décision est prise comme la classe la plus votée.

3.2.2 Boosting

Dans la méthode boosting, on associe des poids aux exemples. Une série de k modèles est itérativement apprise. Après qu'un modèle M_i est construit, les poids des exemples sont mis à jour de telle sorte à attirer l'attention du modèle M_{i+1} aux exemples mal-classées par le modèle M_i. Le Modèle final M_* combine les voltes des k modèles pondérés par leur précisions.

3.3 Classification par analyse des règles d'association

Dans la classification associative (par règles d'association), les règles d'association sont générées et analysées pour les utiliser en classification. L'idée est de rechercher les règles solides contenant dans leur partie droite l'attribut classe, c-à-d de la forme :

$Attribut_1 = v_{att1} \wedge Attribut_2 = v_{att2} \wedge ... \wedge Attribut_n = v_{attn} \Rightarrow Classe = v_{classe}$

Plusieurs études ont montré que cette technique est plus précise que certaines méthodes traditionnelles tel que les arbres de décision.

3.3.1 Algorithme ZeroR

ZeroR est le classifieur le plus simple, il se base sur la classe et ignore tous les attribut. Il prédit tout simplement la classe majoritaire. Il n'a aucune capacité de prédiction mais représente une base de comparaison pour les autres méthodes.

Exemple : "Play Golf = Yes" est le modèle ZeroR pour la base suivante avec une précision de 64%.

Outlook	Temp	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

3.3.2 Algorithme OneR

OneR (One Rule) est un algorithme simple et précis de classification. Il génère une règle pour chaque attribut dans les données puis sélectionne la règle d'erreur minimale comme la seule règle de décision. Pour créer une règle pour un attribut, on construit une table de fréquence pour chaque attribut par rapport à la classe. L'expérience a montré que cet algorithme produit des règles un peu moins précises que plusieurs algorithmes classiques de classification. Cependant, les règles produites sont facilement interprétables par les êtres humains.

Algorithme 2 OneR

pour chaque attribut **faire**

pour chaque valeur de cet attribut **faire** Créer une règle comme suit :

Calculer le nombre d'apparition de chaque valeur de la classe ;

Trouver la classe la plus fréquente ;

Créer une règle assignant cette classe à cette valeur de l'attribut

fin pour

fin pour

Calculer l'erreur totale de chaque règle pour chaque attribut

Choisir l'attribut de l'erreur minimale

Exemple : trouver l'attribut d'erreur minimale en utilisant l'algorithme OneR de la table weather.

Frequences

★		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3

		Play Golf	
		Yes	No
Temp.	Hot	2	2
	Mild	4	2
	Cool	3	1

		Play Golf	
		Yes	No
Humidity	High	3	4
	Normal	6	1

		Play Golf	
		Yes	No
Windy	False	6	2
	True	3	3

IF Outlook = Sunny THEN PlayGolf = Yes
IF Outlook = Overcast THEN PlayGolf = Yes
IF Outlook = Rainy THEN PlayGolf = No

3.3.3 Algorithme CBA

L'un des premiers algorithmes de classification associative est l'algorithme CBA (Classification-Based Association). Il utilise l'algorithme Apriori pour générer les règles d'association puis utilise une heuristique pour construire le classiffieur. Les règles sont ordonnées selon leurs supports et confidences. Si plusieurs règles ont la même partie gauche, la règle de la confidence la plus élevée est utilisée dans le classifieur. Pour classer un nouveau tuplet, la première règle le satisfaisant est utilisée. Le classifieur contient aussi une règle par défaut pour classer les tuplet dont une règles satisfaisante n'existe pas.