Faculté des sciences exactes et informatique

Département d'informatique

Master 1 réseaux

Module : fouille de données

Chapitre 2 : La classification supervisée

I. Introduction :

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).

II. Définitions :

- ***Définition1 :*** **On** dispose d'un ensemble X de N exemples, i.e. des couples (donnée, étiquette). Chaque donnée xi € D est caractérisée par P attributs et par sa classe yi € Y. Dans un problème de classification supervisée, la classe prend sa valeur parmi un ensemble Y fini. Le problème consiste alors, en s'appuyant sur l'ensemble d'exemples X = {**f** (xi; yi) i€{1;….;N} }, à prédire la classe de toute nouvelle donnée x € D.
- ***Définition 2 :***
  Un classeur est une procédure (un algorithme) qui à partir d'un ensemble d'exemples, produit une prédiction de la classe de toute donnée.

Remarque :

- ✓ **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.
- ✓ Concernant le vocabulaire, on utilise le mot étiquette comme synonyme de classe

III. Types de classeur :

On distingue deux grands types de classeurs :

- ✓ Ceux qui utilisent directement les exemples pour prédire la classe d'une donnée ;
- ✓ Ceux pour lesquels on a d'abord construit un modèle et qui, ensuite, utilisent ce modèle pour effectuer leur prédiction.

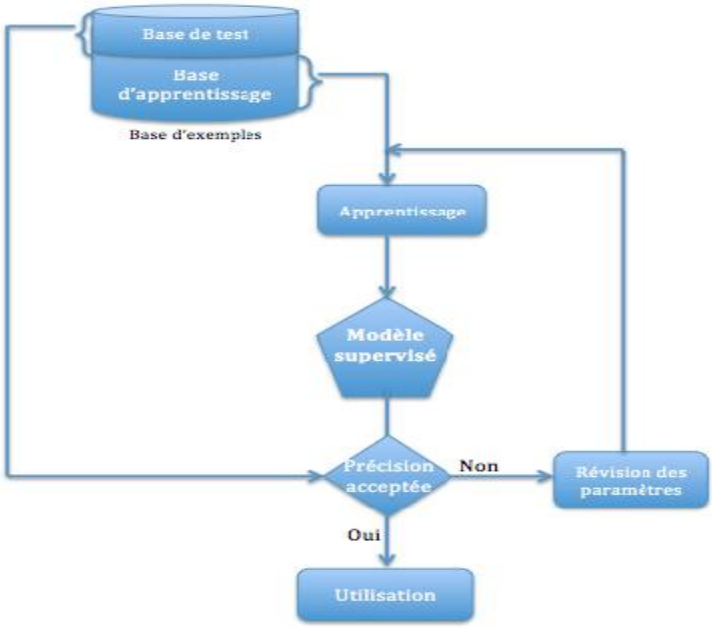
Il ne faut quand même pas tomber dans une vision trop simpliste : il existe naturellement une gradation entre ces deux extrêmes.

Plusieurs problèmes sont à résoudre :

- ✓ Méthode d'induction du classeur ?
- ✓ Comment utiliser le classeur obtenu ?
- ✓ Comment évaluer la qualité du classeur obtenu : taux d'erreur (ou de succès) ?
- ✓ Comment traiter les attributs manquants dans le jeu d'apprentissage ?
- ✓ Comment traiter les attributs manquants dans une donnée à classer ?
- ✓ Estimer la tolérance au bruit : le bruit concerne ici la valeur des attributs de l'exemple avec lequel on construit le classeur.

IV. La classification par modèle :

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 = (x1; x2; ….; xm) 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.

V. 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. Avec

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

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

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

A. La matrice de confusion : 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 ?

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 :

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

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

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

***Table 1** : Matrice de confusion pour la classification binaire*

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.

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

B. Le sur-apprentissage :

Un modèle de décision 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 de 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.

a) **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.

b) **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 D1;D2; ::;Dk de même taille. L'entrainement et le test sont effectués k fois. A l'itération i le sous-ensemble Di 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.

c) **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. La méthode répète le processus k fois et la précision finale P est donnée par :

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

VI. Les méthodes de classification :

i. K plus proche voisins :

L'algorithme des k-plus proches voisins est un des algorithmes de classification les plus simples. Le seul outil dont on a besoin est une distance entre les éléments que l'on veut classifier.

On considère que l'on dispose d'une base d'éléments dont on connaît la classe. On parle de base d'apprentissage, bien que cela soit de l'apprentissage simplifié. Dès que l'on reçoit un nouvel élément que l'on souhaite classifier, on calcule sa distance à tous les éléments de la base. Si cette base comporte 100 éléments, alors on calcule 100 distances et on obtient donc 100 nombres réels. Si k = 25 par exemple, on cherche alors les 25 plus petits nombres parmi ces 100 nombres. Ces 25 nombres correspondent donc aux 25 éléments de la base qui sont les plus proches de l'élément que l'on souhaite classifier. On décide d'attribuer à l'élément à classifier la classe majoritaire parmi ces 25 éléments.

ii. Arbres de décision :

1. **Qu'est-ce qu'un arbre de décision ?:** On se donne un ensemble X de N exemples notés xi dont les P attributs sont quantitatifs ou qualitatifs. Chaque exemple x est étiqueté, c'est-à-dire qu'il lui est associée une classe ou un attribut cible que l'on note y €Y.

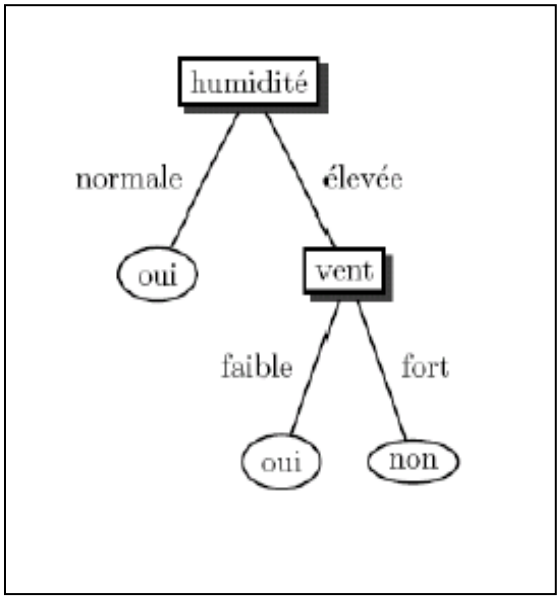
A partir de ces exemples, on construit un arbre dit de décision tel que :

- ✓ chaque nœud correspond à un test sur la valeur d'un ou plusieurs attributs ;
- ✓ chaque branche partant d'un nœud correspond à une ou plusieurs valeurs de ce test ;
- ✓ à chaque feuille est associée une valeur de l'attribut cible.

Un exemple simple :

un exemple de jeu de données : un ensemble de jours (un jour = un exemple), chacun caractérise par un numéro et ses conditions météorologiques (température,

humidité de l'air, force du vent, ciel), l'attribut cible étant jouer au tennis ? , dont les valeurs possibles sont oui et non. Une fois l'arbre de décision construit, on pourra classer une nouvelle donnée pour savoir si on joue ou non ce jour-là ?



Exemple : arbre de décision sur le jeu de données (jouer au tennis)

2. **Génération de l'arbre de décision** : Deux phases dans la génération de l'arbre :
   - ✓ Construction de l'arbre : Arbre peut atteindre une taille élevée
   - ✓ Elaguer l'arbre (Pruning) : Identifier et supprimer les branches qui représentent du "bruit" Améliorer le taux d'erreur

Construction d'un arbre de décision :

La construction d'un arbre de décision optimal est un problème NP-complet, il ne faut pas avoir l'espoir de construire l'arbre de décision optimal pour un jeu d'exemples donne. On va se contenter d'en construire un qui soit correct.

Plusieurs algorithmes ont été proposés, notamment CART dans les années 1980 par Breiman et al. [1984]. Et l'algorithme ID3 de R. Quinlan propose en 1986 qui a été raffiné par la suite (C4.5 puis C5) (cf. Quinlan [1993]). On constate expérimentalement que ces algorithmes sont très performants : ils construisent rapidement des arbres de décision qui prédisent avec une assez grande fiabilité la classe de nouvelles données.

Principe de construction d'un arbre de décision par l'algorithme ID3 : Les tests placés dans un nœud par l'algorithme ID3 concernent exclusivement le test de la valeur d'un et seul . ID3 fonctionne récursivement : il détermine un attribut à placer en racine de l'arbre. Cette racine possède autant de branches que cet attribut prend de valeurs. A chaque branche est associé un ensemble d'exemples dont l'attribut prend la valeur qui étiqueté cette branche ; on accroche alors au bout de cette branche l'arbre de décision construit sur ce sous-ensemble des exemples et en considérant tous les attributs exceptés celui qui vient d'être mis à la racine. Par cette procédure,

l'ensemble des exemples ainsi que l'ensemble des attributs diminuent petit a petit au long de la descente dans l'arbre.

Il reste à résoudre une question centrale ***: quel attribut placer en racine*** ? Une fois cette question résolue, on itérera le raisonnement pour les sous-arbres. L'intuition de la réponse à cette question est que l'on tente de réduire l'hétérogénéité a chaque nœud : les données qui atteignent un certain nœud de l'arbre de décision doivent être plus homogènes que les données atteignant un nœud ancêtre. Pour cela, on a besoin de pouvoir l'homogénéité d'un ensemble de données. En physique, on mesure l'homogénéité par l'entropie. Pour cela, nous avons besoin de définir quelques notions. Commençons par l'entropie introduite initialement par Shannon [1948], notion héritée de la thermodynamique :

Définition1 : Soit un ensemble X d'exemples dont une proportion p+ est positifs et une proportion p-sont négatifs. L'entropie de X est :

$$H(\mathcal{X}) = -p_+ \log_2 p_+ - p_- \log_2 p_-$$

La définition précédente de l'entropie se généralise aisément à un attribut pouvant prendre plus de deux valeurs distinctes :

$$H(\mathcal{X}) = -\sum_{i=1}^{i=n} p_i \log_2 p_i$$

Définition 2 : Soit une population d'exemples X. Le gain d'information de X par rapport à un attribut aj donnée est la variation d'entropie causée par la partition de X selon aj :

$$Gain\ (\mathcal{X}, a_j) = H(\mathcal{X}) - \sum_{v \in\ valeurs(a_j)} \frac{|\mathcal{X}_{a_j=v}|}{|\mathcal{X}|} H(\mathcal{X}_{a_j=v})$$

Le principe de l'algorithme ID3 pour déterminer l'attribut à placer à la racine de l'arbre de décision peut maintenant être exprime : rechercher l'attribut qui possède le gain d'information maximum, le placer en racine, et itérer pour chaque fils, c'est-à-dire pour chaque valeur de l'attribut. Cela étant dit, on peut donner l'algorithme ID3 :

```
ID3
Entrées : ensemble d'attributs A; échantillon E; classe c
Début : initialisé à l'arbre vide;
Si (tous les exemples de E ont la même classe c )
Alors étiqueter la racine par c;
Sinon Si (l'ensemble des attributs A est vide ) alors
        Étiqueter la racine par la classe majoritaire dans E;
        Sinon soit (a le meilleur attribut choisi dans A)
            Étiqueter la racine par a;
            Pour toute valeur v de a
                construire une branche étiquetée par v;
                Soit Eav l'ensemble des exemples tels que e(a) = v;
                ajouter l'arbre construit par ID3(A-{a}, Eav, c);
            Finpour
        Finsinon
Finsinon
Retourner racine;
Fin
```

a) La validation d'un arbre de décision :

Une fois un arbre de décision construit, il est essentiel de le valider en estimant la probabilité que la classe prédite pour une donnée quelconque soit correcte . Dépendant de l'ensemble de données qui est utilisé pour la mesurer, cette quantité est donc une variable aléatoire dont il faut estimer la valeur. Notons que cette section ne concerne pas uniquement les arbres de décision : à quelques détails près, elle peut s'appliquer à tous les algorithmes de classification.

Définition : L'erreur de classification E d'un classeur est la probabilité que ce classeur ne prédise pas correctement la classe d'une donnée de l'espace de données.

Taux de succès Le taux de succès est égal à 1 - E.

Remarque : L'estimation de la qualité de l'arbre de décision construit en tant que classeur de nouvelles données est basée sur les méthodes de validation déjà vu (holdout, validation croisée, bootstarp)

b) La phase d'élagage :

L'élagage consiste à simplifier un arbre de décision en coupant des branches. Il possède deux objectifs :

- ✓ simplifier l'arbre de décision ;
- ✓ diminuer le sur-apprentissage (= augmenter la capacité de généralisation) et, par la même, diminuer le taux d'erreur.

L'élagage peut être effectué avant ou après l'apprentissage, on parle souvent de pré et post-élagage :

– Pré-élagage : effectué lors de la construction de l'arbre, lorsqu'on calcule les caractéristiques statistiques d'une partie des données tel que le gain, on peut décider de l'importance ou non de sa subdivision, et ainsi on coupe complètement des branches qui peuvent être générée.

– **Post-élagage** : effectué après la construction de l'arbre en coupant des sous arbres entiers et en les remplaçant par des feuilles représentant la classe la plus fréquente dans l'ensemble des données de cet arbre. On commence de la racine et on descend, pour chaque nœud interne (non feuille), on mesure sa complexité avant et après sa coupure (son remplacement par une feuille), si la différence est peu importante, on coupe le sous arbre et on le remplace par une feuille.