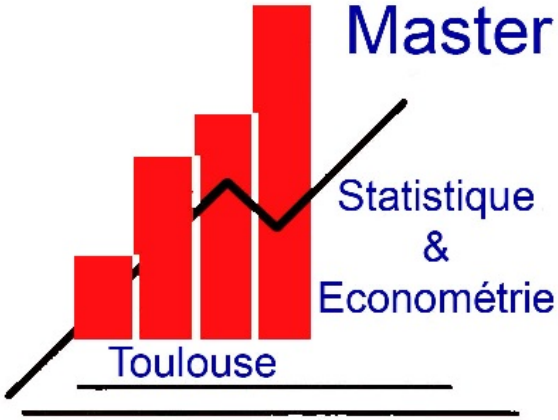

Data Mining 2

Modélisation statistique et apprentissage

E-thème 5 : Arbre de décision, réseaux de neurones

AUTEURS : Philippe Besse
CONTACT : besse@math.ups-tlse.fr
DERNIÈRE MISE À JOUR : 13 septembre 2004
DURÉE : 4 heures
PRE-REQUIS :
Cours de Statistique : estimation, test, régression, de niveau maîtrise
OBJECTIF :
DESCRIPTION :
RÉSUMÉ :
BIBLIOGRAPHIE :
– Besse P. (2000). *Data mining 2. Modélisation statistique et apprentissage*.
MOTS CLÉS :

1 Introduction

Ce chapitre s'intéresse aux méthodes ayant pour objectif la construction d'*arbres binaires*, ou dendogrammes, modélisant une discrimination ou une régression. Complémentaires des méthodes statistiques plus classiques : analyse discriminante, régression linéaire, les solutions obtenues sont présentées sous une forme graphique simple à interpréter, même pour des néophytes, et constituent une aide efficace pour l'aide à la décision. Elles sont basées sur un découpage, par des hyperplans, de l'espace engendré par les variables explicatives. Nommées initialement partitionnement récursif ou segmentation, les développements importants de Breiman et col. (1984) les ont fait connaître sous l'acronyme de CART : Classification and Regression Tree ou encore de C4.5 (Quinlan, 1993) dans la communauté informatique. L'acronyme correspond à deux situations bien distinctes selon que la variable à expliquer, modéliser ou prévoir est qualitative (discrimination ou en anglais *classification*) ou quantitative (régression).
Ces méthodes ne sont efficaces que pour des tailles d'échantillons importantes et elles sont très calculatoires. Les deux raisons : modèle graphique de décision simple à interpréter, puissance de calcul nécessaire, suffisent à expliquer leur popularité récente. De plus, elles requièrent plutôt moins d'hypothèses que des méthodes statistiques classiques et semblent particulièrement adaptées au cas où les variables explicatives sont nombreuses. En effet, la procédure de sélection des variables est intégrée à l'algorithme construisant l'arbre, d'autre part, les interactions sont prises en compte. Néanmoins, cet algorithme suivant une stratégie pas à pas hiérarchisée, il peut, comme dans le cas du choix de modèle en régression, passer à coté d'un optimum global. Ceci souligne encore l'importance de confronter plusieurs approches sur les mêmes données.

2 Construction d'un arbre binaire

2.1 Principe

Les données sont constituées de l'observation de p variables quantitatives ou qualitatives explicatives X^j et d'une variable à expliquer Y qualitative à m modalités $\{\mathcal{T}_\ell; \ell = 1 \ldots, m\}$ ou quantitative réelle, observées sur un échantillon de n individus.
La construction d'un arbre de discrimination binaire (cf. figure 2.1) consiste à déterminer une séquence de *nœuds*.
– Un nœud est défini par le choix conjoint d'une variable parmi les explicatives et d'une *division* qui induit une partition en deux classes. Implicitement, à chaque nœud correspond donc un sous-ensemble de l'échantillon auquel est appliquée une dichotomie.
– Une division est elle-même définie par une valeur seuil de la variable quantitative sélectionnée ou un partage en deux groupes des modalités si la variable est qualitative.
– À la racine ou nœud initial correspond l'ensemble de l'échantillon ; la procédure est ensuite itérée sur chacun des sous-ensembles.
L'algorithme considéré nécessite :

1. la définition d'un critère permettant de sélectionner la "meilleure" division parmi toutes celles *admissibles* pour les différentes variables ;
2. une règle permettant de décider qu'un nœud est terminal : il devient ainsi une *feuille* ;
3. l'affectation de chaque feuille à l'une des classes ou à une valeur de la variable à expliquer.

Le point (ii) est le plus délicat. Il correspond encore à la recherche d'un modèle parcimonieux. Un arbre trop détaillé, associé à une surparamétrisation, est instable et donc probablement plus défaillant pour la prévision d'autres observations. La contribution majeure de Breiman et col. (1984) est justement une stratégie de recherche d'arbre optimal. Elle consiste à

1. construire l'arbre maximal $A_{\max}$,
2. ordonner les sous-arbres selon une séquence emboîtée suivant la décroissance d'un critère pénalisé de déviance ou de taux de mal-classés,

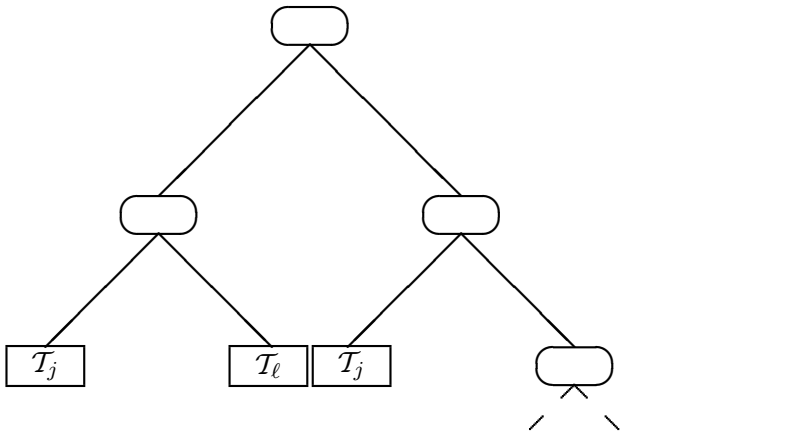

FIG. 1 – Exemple élémentaire d'arbre de classification.

3. puis à sélectionner le sous-arbre optimal ; c'est la procédure d'*élagage*.

Tous ces points sont détaillés ci-dessous.

2.2 Critère de division

Une division est dite *admissible* si aucun des deux nœuds descendants qui en découlent n'est vide. Si la variable explicative est qualitative ordinale avec m modalités, elle fournit $(m-1)$ divisions binaires admissibles. Si elle est seulement nominale le nombre de divisions passe à $2^{(m-1)}-1$. Une variable quantitative se ramène au cas ordinal.
Le critère de division repose sur la définition d'une fonction d'*hétérogénéité* ou de désordre explicitée dans la section suivante. L'objectif étant de partager les individus en deux groupes les plus homogènes au sens de la variable à expliquer. L'hétérogénéité d'un nœud se mesure par une fonction non négative qui doit être

1. nulle si, et seulement si, le nœud est homogène : tous les individus appartiennent à la même modalité ou prennent la même valeur de Y.
2. Maximale lorsque les valeurs de Y sont équiprobables ou très dispersées.

La division du nœud k crée deux fils, gauche et droit. Pour simplifier, ils sont notés $(k+1)$ et $(k+2)$ mais une re-numérotation est nécessaire pour respecter la séquence de sous-arbres qui sera décrite dans la section suivante.
Parmi toutes les divisions admissibles du nœud k, l'algorithme retient celle qui rend la somme $D_{(k+1)} + D_{(k+2)}$ des désordres des nœuds fils minimales. Ceci revient encore à résoudre à chaque étape k de construction de l'arbre :

$$\max_{\{\textit{divisions de} X^j ; j=1,p\}} D_k - (D_{(k+1)} + D_{(k+2)})$$

Graphiquement, la longueur de chaque branche peut être représentée proportionnellement à la réduction de l'hétérogénéité occasionnée par la division.

2.3 Règle d'arrêt

La croissance de l'arbre s'arrête à un nœud donné, qui devient donc terminal ou *feuille*, lorsqu'il est homogène c'est-à-dire lorsqu'il n'existe plus de partition admissible ou, pour éviter un découpage inutilement fin, si le nombre d'observations qu'il contient est inférieur à une valeur seuil à choisir en général entre 1 et 5.

2.4 Affectation

Dans le cas Y quantitative, à chaque feuille est associée une valeur : la moyenne des observations associées à cette feuille. Dans le cas qualitatif, chaque feuille ou nœud terminal est affecté à une classe $\mathcal{T}_\ell$ de Y en considérant le mode conditionnel :

– celle la mieux représentée dans le nœud et il est ensuite facile de compter le nombre d'objets mal classés ;
– la classe *a posteriori* la plus probable au sens bayésien si des probabilités *a priori* sont connues ;
– la classe la moins coûteuse si des coûts de mauvais classement sont donnés.

3 Critères d'homogénéité

Deux cas sont à considérer.

3.1 Y quantitative

On considère le cas plus général d'une division en J classes. Soit n individus et une partition en J classes de tailles $n_j; j = 1, \ldots, J$ avec $n = \sum_{j=1}^{J} n_j$. On numérote $i = 1, \ldots, n_j$ les individus de la $j$ème classe. Soit μ_{ij} (resp.y_{ij}) la valeur "théorique" (resp. l'observation) de Y sur l'individu (i, j) : le $i$ème de la $j$ème classe. L'hétérogénéité de la classe j est définie par :

$$D_j = \sum_{i=1}^{n_j} (\mu_{ij} - \mu_{.j})^2 \quad \text{avec} \quad \mu_{.j} = \sum_{i=1}^{n_j} \mu_{ij}.$$

L'hétérogénéité de la partition est définie par :

$$D = \sum_{j=1}^{J} D_j = \sum_{j=1}^{J} \sum_{i=1}^{n_j} (\mu_{ij} - \mu_{.j})^2;$$

c'est l'inertie intra (homogène à la variance intraclasse) qui vaut $D = 0$ si et seulement si $\mu_{ij} = \mu_{.j}$ pour tout i et tout j.
La différence d'hétérogénéité entre l'ensemble non partagé et l'ensemble partagé selon la partition J est

$$\begin{aligned} \Delta &= \sum_{j=1}^{J} \sum_{i=1}^{n_j} (\mu_{ij} - \mu_{..})^2 - \sum_{j=1}^{J} \sum_{i=1}^{n_j} (\mu_{ij} - \mu_{.j})^2 \text{ où } \mu_{..} = \frac{1}{n} \sum_{j=1}^{J} \sum_{i=1}^{n_j} \mu_{ij}. \\ &= \sum_{j=1}^{J} n_j (\mu_{..} - \mu_{.j})^2; \end{aligned}$$

c'est encore homogène à la variance inter classe ou "désordre" des barycentres qui vaut $\Delta = n_1 n_2((\mu_{.1} - \mu_{.2})^2$ pour $J = 2$ dans le cas qui nous intéresse.
L'objectif, à chaque étape, est de maximiser Δ c'est-à-dire de trouver la variable induisant une partition en 2 classes associée à une inertie (variance) intraclasse minimale ou encore qui rend l'inertie (la variance) interclasse la plus grande.
Les quantités sont estimées :

$$D_j \quad \text{par} \quad \widehat{D_j} = \sum_{i=1}^{n_j} (y_{ij} - y_{.j})^2 \tag{1}$$

$$D \quad \text{par} \quad \widehat{D} = \sum_{j=1}^{J} \widehat{D_j} = \sum_{j=1}^{J} \sum_{i=1}^{n_j} (y_{ij} - y_{.j})^2. \tag{2}$$

Sous hypothèse gaussienne :

$$Y_{ij} = \mu_{.j} + u_{ij} \quad \text{avec} \quad + u_{ij} \sim \mathcal{N}(0, \sigma^2),$$

la log-vraisemblance

$$\log L = \text{Cste} - \frac{n}{2}\log(\sigma^2) - \frac{1}{2\sigma^2}\sum_{j=1}^{J}\sum_{i=1}^{n_j}(y_{ij} - \mu_{.j})^2$$

est rendue maximale pour

$$\mathcal{L}_\mu = \sup_\mu \log L = \text{Cste} - \frac{n}{2}\log(\sigma^2) - \frac{1}{2\sigma^2}\sum_{j=1}^{J}\sum_{i=1}^{n_j}(y_{ij} - y_{.j})^2.$$

Pour le modèle saturé (une classe par individu) : $y_{ij} = \mu_{ij} + u_{ij}$, cet optimum devient :

$$\mathcal{L}_s = \sup_\mu \log L = \text{Cste} - \frac{n}{2}\log(\sigma^2).$$

La déviance (par rapport au modèle saturé) s'exprime alors comme :

$$\mathcal{D}_\mu = 2\sigma^2(\mathcal{L}_s - \mathcal{L}_\mu) = \widehat{D}.$$

Le raffinement de l'arbre est donc associé à une décroissance, la plus rapide possible, de la déviance. C'est l'optique retenue dans le logiciel Splus. On peut encore dire que la division retenue est celle qui rend le test de Fisher (analyse de variance), comparant les moyennes entre les deux classes, le plus significatif possible.

3.2 Y qualitative

Dans ce cas, la fonction d'hétérogénéité, ou de désordre d'un nœud, est définie à partir de la notion d'entropie, du critère de concentration de Gini ou encore d'une statistique de test du χ^2. En pratique, il s'avère que le choix du critère importe moins que celui du niveau d'élagage. Le premier critère (entropie) est souvent préféré (Splus) car il s'interprète encore comme un terme de déviance mais d'un modèle multinomial cette fois.
On considère une variable à expliquer qualitative, Y à m modalités ou catégories $\mathcal{T}$ numérotées $\ell = 1, \ldots, m$. L'arbre induit une partition pour laquelle n_{+k} désigne l'effectif de la $k$ème classe ou $k$ème nœud. Soit

$$p_{\ell k} = P[\mathcal{T}_\ell \mid k] \quad \text{avec} \quad \sum_{\ell=1}^{m} p_{\ell k} = 1$$

la probabilité qu'un élément du $k$ème nœud appartienne à la ℓème classe.
Le *désordre* du $k$ème nœud, défini à partir de l'entropie, s'écrit avec la convention $0\log(0) = 0.$:

$$D_k = -2\sum_{\ell=1}^{m} n_{+k} p_{\ell k}\log(p_{\ell k})$$

tandis que l'hétérogénéité ou désordre de la partition est encore :

$$D = \sum_{k=1}^{K} D_k = -2\sum_{k=1}^{K}\sum_{\ell=1}^{m} n_{+k} p_{\ell k}\log(p_{\ell k}).$$

Cette quantité est positive ou nulle, elle est nulle si et seulement si les probabilités $p_{\ell k}$ ne prennent que des valeurs 0 sauf une égale à 1 correspondant à l'absence de mélange.
Désignons par $n_{\ell k}$ l'effectif observé de la ℓème classe dans le $k$ème nœud. Un nœud k de l'arbre représente un sous-ensemble de l'échantillon d'effectif $n_{+k} = \sum_{\ell=1}^{m} n_{\ell k}$.

Les quantités sont estimées :

$$D_k \quad \text{par} \widehat{D_k} \quad = \quad -2\sum_{\ell=1}^{m} n_{+k}\frac{n_{\ell k}}{n_{+k}}\log\frac{n_{\ell k}}{n_{+k}} \tag{3}$$

$$D \quad \text{par} \quad \widehat{D} \quad = \quad \sum_{k=1}^{K}\widehat{D_k} = -2\sum_{k=1}^{K}\sum_{\ell=1}^{m} n_{\ell k}\log\frac{n_{\ell k}}{n_{+k}}. \tag{4}$$

Considérons, pour chaque classe ou nœud k, un modèle multinomial à m catégories de paramètre :

$$p_k = (p_{1k}, \ldots, p_{mk}), \quad \text{avec} \quad \sum_{\ell=1}^{m} p_{\ell k} = 1.$$

Pour ce modèle, la logvraisemblance :

$$\log L = \text{Cste} + \sum_{k=1}^{K}\sum_{\ell=1}^{m} n_{\ell k}\log(p_{\ell k})$$

est rendue maximale pour

$$\mathcal{L}_\mu = \sup_{p_{\ell k}} \log L = \text{Cste} + \sum_{k=1}^{K}\sum_{\ell=1}^{m} n_{\ell k}\log\frac{n_{\ell k}}{n_{+k}}.$$

Pour le modèle saturé (une catégorie par objet), cet optimum prend la valeur de la constante et la déviance (par rapport au modèle saturé) s'exprime comme :

$$\mathcal{D} = -2\sum_{k=1}^{K}\sum_{\ell=1}^{m} n_{\ell k}\log\frac{n_{\ell k}}{n_{+k}} = \widehat{D}.$$

Comme pour l'analyse discriminante décisionnelle, les probabilités conditionnelles sont définies par la règle de Bayes lorsque les probabilités *a priori* π_ℓ d'appartenance à la ℓème classe sont connues. Dans le cas contraire, les probabilités de chaque classe sont estimées sur l'échantillon et donc les probabilités conditionnelles s'estiment simplement par des rapports d'effectifs : $p_{\ell k}$ est estimée par $n_{\ell k}/n_{+k}$. Enfin, il est toujours possible d'introduire, lorsqu'ils sont connus, des coûts de mauvais classement et donc de se ramener à la minimisation d'un risque bayésien.

4 Élagage

Dans des situations complexes, la démarche proposée conduit à des arbres extrêmement raffinés et donc à des modèles de prévision très instables car fortement dépendants des échantillons qui ont permis leur estimation. On se trouve donc dans une situation de sur-ajustement à éviter au profit de modèles plus parcimonieux donc plus robuste au moment de la prévision. Cet objectif est obtenu par une procédure d'*élagage* (*pruning*) de l'arbre.

Le principe de la démarche, introduite par Breiman et col. (1984), consiste à construire une suite emboîtée de sous-arbres de l'arbre maximum par élagage successif puis à choisir, parmi cette suite, l'arbre optimal au sens d'un critère. La solution ainsi obtenue par un algorithme pas à pas n'est pas nécessairement globalement optimale mais l'efficacité et la fiabilité sont préférées à l'optimalité.

4.1 Construction de la séquence d'arbres

Pour un arbre A donné, on note K le nombre de feuilles ou nœuds terminaux de A ; la valeur de K exprime la complexité de A. La mesure de qualité de discrimination d'un arbre A s'exprime par un critère

$$D(A) = \sum_{k=1}^{K} D_k(A)$$

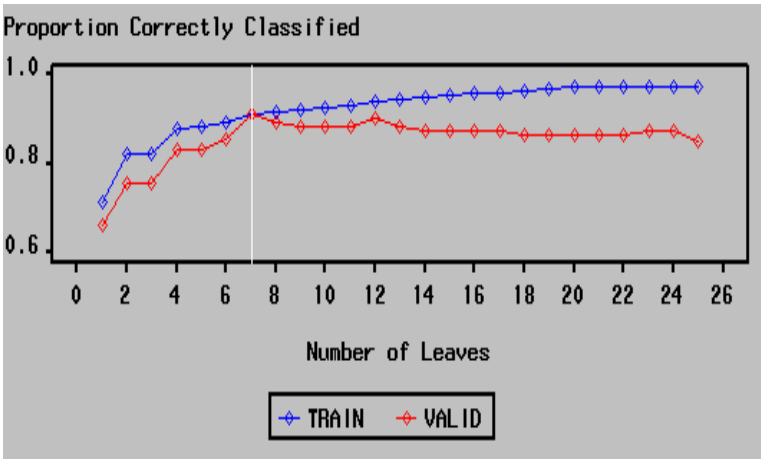

FIG. 2 – *Carte Visa : choix du nombre de feuilles par échantillon de validation (SEM, 2001).*

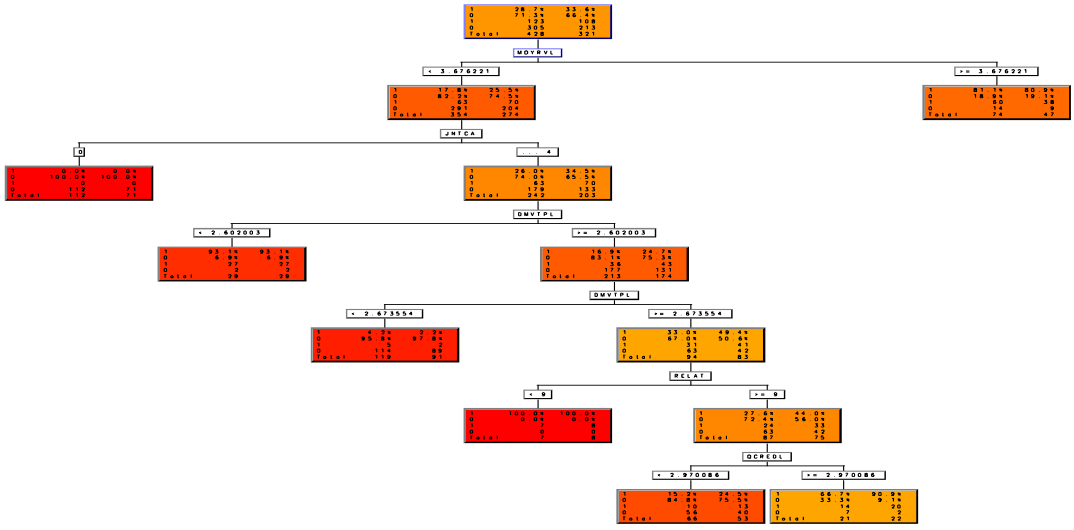

FIG. 3 – *Carte Visa : arbre de décision élagué suivant l'échantillon de validation(SEM, 2001).*

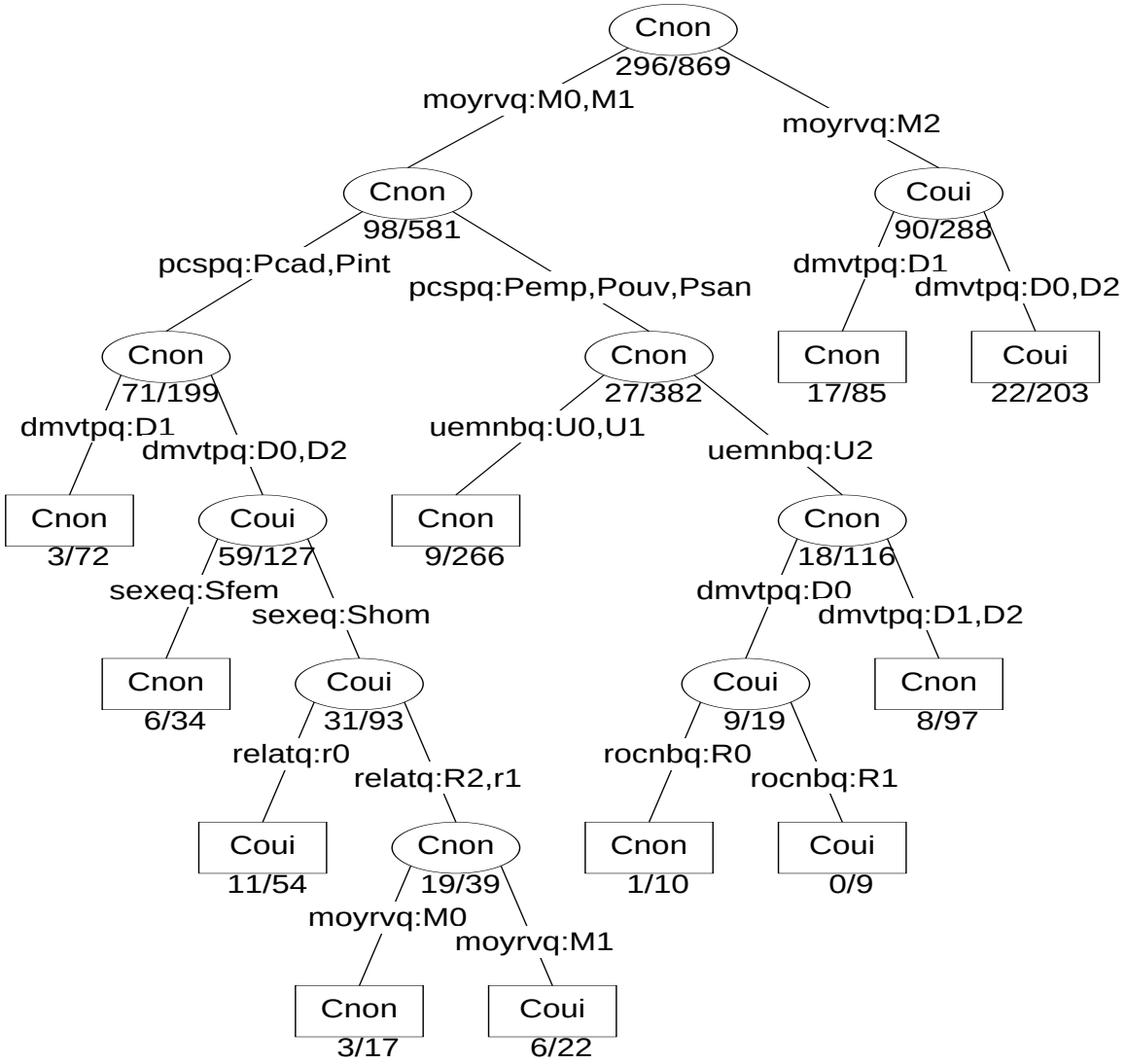

FIG. 4 – *Carte Visa : arbre de décision (Splus, 1993) élagué par validation croisée.*

où $D_k(A)$ est le nombre de mal classés ou la déviance ou le coût de mauvais classement de la $k$ème feuille de l'arbre A.
La construction de la séquence d'arbres emboîtés repose sur une pénalisation de la complexité de l'arbre :

$$C(A) = D(A) + \gamma K.$$

Pour $\gamma = 0$, $A_{\max} = A_K$ minimise $C(A)$. En faisant croître γ, l'une des divisions de A_K, celle pour laquelle l'amélioration de D est la plus faible (inférieure à γ), apparaît comme superflue et les deux feuilles obtenues sont regroupées (élaguées) dans le nœud père qui devient terminal ; A_K devient A_{K-1}.
Le procédé est itéré pour la construction de la séquence emboîtée :

$$A_{\max} = A_K \supset A_{K-1} \supset \cdots A_1$$

où A_1, le nœud racine, regroupe l'ensemble de l'échantillon.
Un graphe représente la décroissance ou éboulis de la déviance (ou du taux de mal classés) en fonction du nombre croissant de feuilles dans l'arbre ou, c'est équivalent, en fonction de la valeur décroissante du coefficient de pénalisation γ.

4.2 Recherche de l'arbre optimal

Les procédures d'élagage diffèrent par la façon d'estimer l'erreur de prédiction. Le graphe précédemment obtenu peut se lire comme un éboulis de valeur propre. Quand l'amélioration du critère est jugé trop petite ou négligeable, on élague l'arbre au nombre de feuilles obtenues. L'évaluation de la déviance ou du taux de mauvais classement estimée par resubstitution sur l'échantillon d'apprentissage est biaisée (trop optimiste). Une estimation sans biais est obtenue par l'utilisation d'un autre échantillon (validation) ou encore par validation croisée. La procédure de validation croisée présente dans ce cas une particularité car la séquence d'arbres obtenue est différente pour chaque estimation sur l'un des sous-échantillons. L'erreur moyenne n'est pas, dans ce cas, calculée pour chaque sous-arbre avec un nombre de feuilles donné mais pour chaque sous-arbre correspondant à une valeur fixée du coefficient de pénalisation. À la valeur de γ minimisant l'estimation de l'erreur de prévision, correspond ensuite l'arbre jugé optimal dans la séquence estimée sur tout l'échantillon d'apprentissage.
Le principe de sélection d'un arbre optimal est donc décrit dans l'algorithme ci-dessous.
algorithme : Sélection d'arbre
– Construction de l'arbre maximal $A_{\max}$.
– Construction de la séquence $A_K \ldots A_1$ d'arbres emboîtés.
– Estimation sans biais (échantillon de validation ou validation croisée) des déviances $D(A_K), \ldots, D(A_1)$.
– Représentation de $D(A_k)$ en fonction de k ou de γ.
– Choix de k rendant $D(A_k)$ minimum.

5 Introduction aux méthodes connexionistes

Nous nous intéressons ici à une branche de l'Informatique fondamentale qui, sous l'appellation d'*Intelligence Artificielle*, a pour objectif de simuler des comportements du cerveau humain. Les premières tentatives de modélisation du cerveau sont anciennes et précèdent même l'ère informatique. C'est en 1943 que Mc Culloch (neurophysiologiste) et Pitts (logicien) ont proposé les premières notions de *neurone formel*. Ce concept fut ensuite mis en réseau avec une couche d'entrée et une sortie par Rosenblatt en 1959 pour simuler le fonctionnement rétinien et tacher de reconnaître des formes. C'est l'origine du *perceptron*. Cette approche dite *connexioniste* a atteint ses limites technologiques, compte tenu de la puissance de calcul de l'époque, mais aussi théoriques au début des années 70.
L'approche connexioniste à *connaissance répartie* a alors été supplantée par l'approche *symbolique* ou séquentielle qui promouvait les *systèmes experts* à *connaissance localisée*. L'objectif était alors d'automatiser le principe de l'expertise humaine en associant trois concepts :

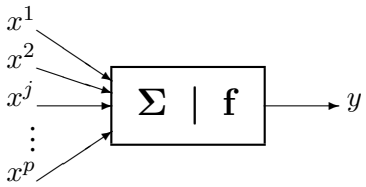

FIG. 5 – Représentation d'un neurone formel.

– une *base de connaissance* dans laquelle étaient regroupées "toutes" les connaissances d'experts humains sous forme de propositions logiques élémentaires ou plus élaborées en utilisant des quantificateurs (logique du premier ordre).
– une *base de faits* contenant les observations du cas à traiter comme, par exemple, des résultats d'examens, d'analyses de sang, de salive pour des applications biomédicales de choix d'un antibiotique,
– un *moteur d'inférence* chargé d'appliquer les règles expertes sur la base de faits afin d'en déduire de nouveaux faits jusqu'à la réalisation d'un objectif comme l'élaboration du traitement d'un infection bactérienne.

Face aux difficultés rencontrées lors de la modélisation des connaissances d'un expert humain, au volume considérable des bases de connaissance qui en découlait et au caractère exponentiel de la complexité des algorithmes d'inférence mis en jeu, cette approche s'est éteinte avec les années 80. En effet, pour les systèmes les plus compliqués à base de calcul des prédicats du premier ordre, on a pu montrer qu'ils conduisaient à des problèmes NP complets et donc dont la solution pouvait être atteinte mais pas nécessairement en un temps fini !

L'essor technologique et surtout quelques avancées théoriques :
– algorithme d'estimation par rétropropagation de l'erreur par Hopkins en 1982,
– analogie de la phase d'apprentissage avec les modèles markoviens de systèmes de particules de la mécanique statistique (verres de spin) par Hopfield en 1982,

au début des années 80 ont permis de relancer l'approche connexioniste. Celle-ci a connu au début des années 90 un développement considérable si l'on considère le nombre de publications et de congrès qui lui ont été consacrés mais aussi les domaines d'applications très divers où elle apparaît. Sur de nombreux objectifs, justement ceux propres au data mining, les réseaux neuronaux ne rentrent pas nécessairement en concurrence avec des méthodes statistiques bientôt centenaires mais apportent un point de vue complémentaire qu'il est important de considérer (Thiria et col. 1997).

6 Réseaux de neurones

Un *réseau neuronal* est l'association, en un graphe plus ou moins complexe, d'objets élémentaires, les *neurones formels*. Les principaux réseaux se distinguent par l'organisation du graphe (en couches, complets…), c'est-à-dire leur architecture, son niveau de complexité (le nombre de neurones) et par le type des neurones (leurs fonctions de transition).

6.1 Neurone formel

De façon très réductrice, un neurone biologique est une cellule qui se caractérise par
– des synapses, les points de connexion avec les autres neurones, fibres nerveuses ou musculaires ;
– des dentrites, les "entrées" du neurones ;
– l'axone, la "sortie" du neurone vers d'autres neurones ou fibres musculaires ;
– le noyau qui active la sortie en fonction des stimuli en entrée.

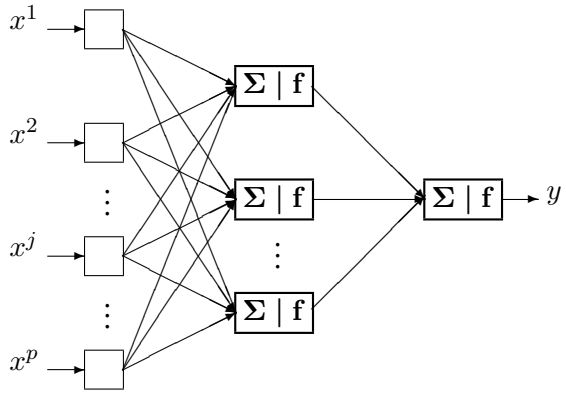

FIG. 6 – Exemple de perceptron multicouche élémentaire avec une couche cachée et une couche de sortie.

Par analogie, le neurone formel est un modèle qui se caractérise par un état interne $s \in \mathcal{S}$, des signaux d'entrée $x_1, \ldots, x_p$ et une fonction de transition d'état

$$s = h(x_1, \ldots, x_p) = f\left(\beta_0 + \sum_{j=1}^{p} \beta_j x_j\right).$$

La fonction de transition opère une transformation d'une combinaison affine des signaux d'entrée, β_0 étant appelé le biais du neurone. Cette combinaison affine est déterminée par un *vecteur de poids* $[\beta_0, \ldots, \beta_p]$ associé à chaque neurone et dont les valeurs sont estimées dans la phase d'apprentissage. Ils constituent "la mémoire" ou "connaissance répartie" du réseau.
Les différents types de neurones se distinguent par la nature f de leur fonction de transition. Les principaux types sont :
– *linéaire* f est la fonction identité,
– *sigmoïde* $f(x) = 1/(1 + e^x)$,
– *seuil* $f(x) = \mathbf{1}_{[0,+\infty[}(x)$,
– *stochastiques* $f(x) = 1$ avec la probabilité $1/(1 + e^{-x/H})$, 0 sinon (H intervient comme une température dans un algorithme de recuit simulé),
– ...
Les modèles linéaires et sigmoïdaux sont bien adaptés aux algorithmes d'apprentissage comme celui de rétropropagation du gradient car leur fonction de transition est différentiable. Ce sont les plus utilisés. Le modèle à seuil est sans doute plus conforme à la "réalité" biologique mais pose des problèmes d'apprentissage. Enfin le modèle stochastique est utilisé pour des problèmes d'optimisation globale de fonctions perturbées ou encore pour les analogies avec les systèmes de particules. On ne le rencontre pas en *data mining*.

7 Perceptron multicouche

7.1 Architecture

Le perceptron multicouche (PMC) est un réseau composé de couches successives. Une *couche* est un ensemble de neurones n'ayant pas de connexion entre eux. Une couche d'entrée lit les signaux entrant,

un neurone par entrée x_j, une couche en sortie fournit la réponse du système. Selon les auteurs, la couche d'entrée qui n'introduit aucune modification n'est pas comptablisée. Une ou plusieurs couches cachées participent au transfert. Un neurone d'une couche cachée est connecté en entrée à chacun des neurones de la couche précédente et en sortie à chaque neurone de la couche suivante.
Un perceptron multicouche réalise donc une transformation

$$y = \phi(x_1, \ldots, x_p; \boldsymbol{\beta})$$

où $\boldsymbol{\beta}$ est le vecteur contenant chacun des paramètres $\beta_{jk\ell}$ de la $j$ème entrée du $k$ème neurone de la ℓème couche ; la couche d'entrée ($\ell = 0$) n'est pas paramétrée, elle ne fait que distribuer les entrées sur tous les neurones de la couche suivante.
Par souci de cohérence, nous avons tâché de conserver les mêmes notations à travers les différents chapitres. Ainsi, les *entrées* d'un réseau sont encore notées $x^1, \ldots, x^p$ comme les variables explicatives d'un modèle tandis que les *poids* des entrées sont des paramètres $\boldsymbol{\beta}$ à estimer lors de la procédure d'*apprentissage* et que la *sortie* est la variable à expliquer ou cible du modèle.

7.2 Apprentissage

Supposons que l'on dispose d'une base d'apprentissage de taille n d'observations $(x_i^1, \ldots, x_i^p; y_i)$ des variables explicatives $X^1, \ldots, X^p$ et de la variable à prévoir Y. L'apprentissage est l'estimation $\widehat{\boldsymbol{\beta}}$ des paramètres du modèle solutions du problème des moindres carrés[1] :

$$\widehat{\boldsymbol{\beta}} = \arg\min_{\mathbf{b}} Q(\mathbf{b}) \quad \text{avec} \quad Q(\mathbf{b}) = \frac{1}{n}\sum_{i=1}^{n}[y_i - \phi(x_i^1, \ldots, x_i^p; (b))]^2.$$

L'algorithme d'optimisation le plus utilisé est celui de rétropropagation du gradient basé sur l'idée suivante : en tout point $\mathbf{b}$, le vecteur gradient de Q pointe dans la direction de l'erreur croissante. Pour faire décroître Q il suffit donc de se déplacer en sens contraire. Il s'agit d'un algorithme itératif modifiant les poids de chaque neurone selon :

$$b_{jk\ell}(i) = b_{jk\ell}(i-1) + \Delta b_{jk\ell}(i)$$

où la correction $\Delta b_{jk\ell}(i)$ est proportionnelle au gradient et à l'erreur attribuée à l'entrée concernée $\varepsilon_{jk\ell}(i)$ et incorpore un terme d'"inertie" $\alpha b_{jk\ell}(i-1)$ permettant d'amortir les oscillations du système :

$$\Delta b_{jk\ell}(i) = -\tau\varepsilon_{jk\ell}(i)\frac{\partial Q}{\partial b_{jk\ell}} + \alpha b_{jk\ell}(i-1).$$

Le coefficient de proportionnalité τ est appelé le *taux d'apprentissage*. Il peut être fixe à déterminer par l'utilisateur ou encore varier en cours d'exécution selon certaines règles paramétrées par l'utilisateur. Il paraît en effet intuitivement raisonnable que, grand au début pour aller plus vite, ce taux décroisse pour aboutir à un réglage plus fin au fur et à mesure que le système s'approche d'une solution. La formule de *rétropropagation de l'erreur* fournit, à partir des erreurs observées sur les sorties, l'expression de l'erreur attribuée à chaque entrée de la couche de sortie à la couche d'entrée.
La littérature sur le sujet propose quantités de recettes destinées à améliorer la vitesse de convergence de l'algorithme ou bien lui éviter de rester collé à une solution locale défavorable. Des propriétés (dynamique markovienne ergodique et convergence vers la mesure stationnaire) de cet algorithme impliquent une convergence presque sûre ; la probabilité d'atteindre une précision fixée *a priori* tend vers 1 lorsque la taille de l'échantillon d'apprentissage tend vers l'infini.
Une amélioration importante consiste à introduire une terme de pénalisation ou régularisation comme en *ridge* dans le critère à optimiser. Celui-ci devient alors :

$$\widehat{\boldsymbol{\beta}} = \arg\min_{\mathbf{b}} Q(\mathbf{b}) + \delta\|b\|^2.$$

[1] Équivalent à une maximisation de la vraisemblance dans le cas gaussien.

Le paramètre δ (*decay*) doit être fixé par l'utilisateur ; plus il est important et moins les paramètres ou poids peuvent prendre des valeurs "cahotiques" contribuant ainsi à limiter les risques de surapprentissage.
algorithme : Rétropropagation du gradient
– `Initialisation`
  – Les poids $b_{jk\ell}$ par tirage aléatoire selon une loi uniforme sur $[0, 1]$.
  – Normaliser dans $[0, 1]$ les données d'apprentissage.
– `Tant que` $Q >$ errmax ou niter$<$itermax.
– Ranger la base d'apprentissage dans un nouvel ordre aléatoire.
  – `Pour` chaque élément $i = 1, \ldots, n$ de la base `Faire`
    – Calculer $\varepsilon(i) = y_i - \phi(x_i^1, \ldots, x_i^p; (b)(i-1))$ en propageant les entrées vers l'avant.
    – L'erreur est "rétropropagée" dans les différentes couches afin d'affecter à chaque entrée une responsabilité dans l'erreur globale.
    – Mise à jour de chaque poids $b_{jk\ell}(i) = b_{jk\ell}(i-i) + \Delta b_{jk\ell}(i)$
  – `Fin Pour`
– `Fin Tant que`

7.3 Utilisation

On pourra se reporter à l'abondante littérature sur le sujet (Haykin, 1994) pour obtenir des précisions sur les algorithme d'apprentissage et leurs nombreuses variantes. Il est important de rappeler la liste des choix qui sont laissés à l'utilisateur. En effet, même si les logiciels proposent des valeurs par défaut, il est fréquent que cet algorithme connaisse quelques soucis de convergence.
L'utilisateur doit donc déterminer

1. les variables d'entrée et la variable de sortie ; leur faire subir comme pour toutes méthodes statistiques, d'éventuelles transformations.
2. L'architecture du réseau : le nombre de couches cachées (en général une ou deux) qui correspond à une aptitude à traiter des problèmes de non-linéarité, le nombre de neurones par couche cachée. Ces deux choix conditionnent directement le nombre de paramètres (de poids) à estimer. Ils participent à la recherche d'un bon compromis biais/variance c'est-à-dire à l'équilibre entre qualité d'apprentissage et qualité de prévision. À la louche, on considère en pratique qu'il faut un échantillon d'apprentissage au moins dix fois plus grand que le nombre de paramètres à estimer.
3. Trois autres paramètres interviennent également sur ce compromis : le nombre maximum d'itérations, l'erreur maximum tolérée et un terme éventuel de régularisation (*decay*). En renforçant ces critères on améliore la qualité de l'apprentissage ce qui peut se faire au détriment de celle de la prévision.
4. Le taux d'apprentissage ainsi qu'une éventuelle stratégie d'évolution de celui-ci.

En pratique, tous ces paramètres ne sont pas réglés simultanément par l'utilisateur. Celui-ci est confronté à des choix concernant principalement le contrôle du sur-apprentissage ; choix du paramètre : limiter le nombre de neurones ou la durée d'apprentissage ou encore augmenter le coefficient de pénalisation de la norme des paramètres ; choix du mode d'estimation de l'erreur : échantillon test, validation croisée ou bootstrap. Ces choix sont souvent pris par défaut dans la plupart des logiciels commerciaux. Il est important d'en connaître les implications.
Le nombre de couches reste restreint. On montre en effet que toute fonction que toute fonction continue d'un compact de $\mathbb{R}^P$ dans $\mathbb{R}^q$ peut être approchée avec une précision arbitraire par un réseau à une couche cachée en adaptant le nombre de neurones. Le contrôle de la complexité du modèle ou plus généralement d'un sur-apprentissage peut se faire à l'aide de plusieurs paramètres : le nombre de neurones, une pénalisation de la norme du vecteur des poids ou paramètres comme en *ridge* (régularisation) ou encore par la durée de l'apprentissage. Ces paramètres sont optimisés en considérant un échantillon de validation et le plus simple consiste à arrêté l'apprentissage lorsque l'erreur sur l'échantillon de validation commence à se dégrader tandis que celle sur l'échantillon d'apprentissage ne peut que continuer à décroître.

Les champs d'application des PMC sont très nombreux : discrimination, prévision d'une série temporelle, reconnaissance de forme... Ils sont en général bien explicités dans les documentations des logiciels spécialisés.
Les critiques principales énoncées à l'encontre du PMC concernent les difficultés liés à l'apprentissage (temps de calcul, taille de l'échantillon, localité de l'optimum obtenu) ainsi que son statut de boîte noir. En effet, contrairement à un modèle de discrimination ou un arbre, il est *a priori* impossible de connaître l'influence effective d'une entrée (une variable) sur le système dès qu'une couche cachée intervient. Néanmoins, des techniques de recherche de sensibilité du système à chacune des entrées permettent de préciser les idées et, éventuellement de simplifier le système en supprimant certaines des entrées.
En revanche, ils possèdent d'indéniables qualités lorsque l'absence de linéarité et/ou le nombre de variables explicatives rendent les modèles statistiques traditionnelles inutilisables. Leur flexibilité alliée à une procédure d'apprentissage intégrant la pondération (le choix) des variables comme de leurs interactions peuvent les rendre très efficaces (Besse et col. 2001).