Apprentissage automatique

François Denis, fdenis@cmi.univ-mrs.fr
Laboratoire d'Informatique Fondamentale de Marseille
LIF - UMR CNRS 6166
Equipe Bases de données et Apprentissage

1 Introduction à l'apprentissage automatique.

1.1 Exemples introductifs

En classification supervisée

- *Email/Spam* : on souhaite concevoir un programme capable de filtrer les messages électroniques en écartant systématiquement les annonces publicitaires et autres messages non sollicités. Les données de travail seront constituées d'un certain nombre de couples (d, c), où d est une représentation d'un message électronique, par exemple un vecteur d'occurrences ou de fréquences de mots, et où c est la classe attribuée au document : *Email* ou *Spam*. On cherche alors une fonction, appelée *classifieur*, capable d'attribuer automatiquement une de ces deux classes à un nouveau document. On souhaite bien entendu que cette nouvelle classification soit le plus souvent identique à ce qu'on aurait pu faire manuellement. Ce problème est un cas particulier de *catégorisation* ou *classification supervisée*. Le terme *supervisé* provient de ce que non seulement les classes sont connues a priori mais aussi, de ce que le classifieur doit être construit à partir de données déjà classées : on suppose l'existence d'un *superviseur* capable d'associer (correctement) une classe à une description.
- *Reconnaissance des visages, des chiffres, ...* : on souhaite écrire un programme capable de reconnaître une personne à partir d'une photo, ou un code postal à partir d'un manuscrit. Les données de travail sont par exemple des chiffres écrits sur une rétine de 16x16 pixels, chacun étant associé à une classe parmi $\{0, 1, \ldots, 9\}$. Il s'agit là encore, dans le domaine de la *reconnaissance des formes* (*pattern recognition*), d'un problème de classification supervisée.

Il s'agit dans tous les cas de construire, à partir d'un échantillon limité d'observations composées de couples (description - classe), une procédure de classification qui sera capable de traiter correctement des nouveaux cas. Le terme *classification* ne doit pas induire en erreur : classer de nouveaux cas, c'est apprendre.

On parle de *classification binaire* lorsqu'il n'y a que deux classes, de *classification n-aire* dans le cas général, et de *classification multi-étiquettes* lorsque plusieurs classes peuvent

être attribuées à une même description (exemple des mots-clés d'un document).

En classification non-supervisée

- On peut souhaiter identifier des profils parmi les clients d'une entreprise, les utilisateurs de transports en commun ou les spectateurs d'une chaîne de télévision : la fameuse ménagère de plus de 40 ans, les bobos, les couples en voie d'acheter un bien immobilier, ... On parle d'*apprentissage non supervisé* ou de *clustering*[1] : il faut déterminer les classes, en même temps qu'une fonction de classification attribuant une classe à un individu.
- Une puce à ADN décrit les niveaux d'expression de quelques milliers de gènes prélevés dans divers tissus sains ou cancéreux. Peut-on en déduire des *patterns d'expression*, c'est-à-dire des ensembles de gènes s'exprimant de manière corrélée et donc susceptibles de définir ensemble une fonction biologique précise ? Les classes, ici les fonctions biologiques, ne sont pas connues au départ et les individus observés ne sont pas directement étiquetés par une ou des fonctions. Il s'agit encore d'apprentissage non supervisé.

En régression supervisée

- On peut souhaiter prévoir la température, la pression atmosphérique ou la vitesse du vent en fonction de divers paramètres numériques ou symboliques. Les données consistent en couples (d, x) description-mesure, la mesure étant le plus souvent décrite par un nombre réel. Il s'agit de problèmes de *régression*[2] *supervisée*.
- On peut souhaiter, dans le problème de détection des SPAMS, associer à un nouveau document la *probabilité* qu'il soit un SPAM. Les données sont toujours discrètes mais la fonction cherchée est à valeurs réelles. On parle encore de problème de régression.

En estimation de densité.

- Lorsqu'on souhaite différencier deux auteurs à partir des documents qu'ils ont produits, une approche naturelle consiste à étudier les fréquences de mots apparaissant dans ces documents en espérant que ces fréquences permettront d'évaluer la probabilité qu'un nouveau document ait été produit par l'un ou l'autre auteur.
- De même lorsqu'on souhaite déterminer si une séquence d'ADN code ou non pour un gène.

[1] Le terme de *classification* employé seul a des usages différents selon les communautés : pour les statisticiens, il désigne par défaut la classification non supervisée alors que dans la communauté de l'apprentissage automatique, il désignera plutôt la classification supervisée.

[2] "Ce terme de régression provient des travaux du statisticien Galton qui étudiait la taille des enfants (Y) en fonction de la taille de leur père (X). Il avait constaté expérimentalement que la taille moyenne des fils dont le père avait une taille supérieure à la moyenne $E(X)$ était elle-même supérieure à $E(Y)$ mais dans une moindre mesure ; il y avait donc *régression* au sens ordinaire du mot" [Sap90].

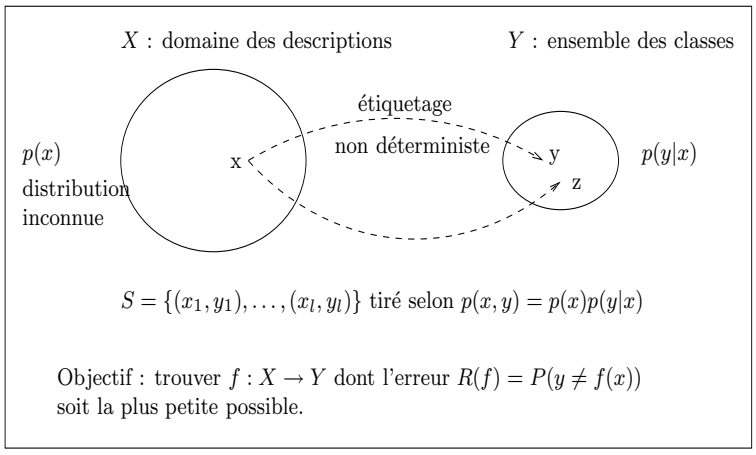
Autrement dit, on suppose que les descriptions ne sont pas produites selon les mêmes lois dans une classe et dans l'autre et on cherche à estimer les lois des distributions ou densités correspondantes. On parle alors d'*estimation de densité*.

Quelques ouvrages de référence

- *Apprentissage artificiel*, d'Antoine Cornuejols et Laurent Miclet, est un livre récent, très complet et très pédagogique [CM02]. Il a aussi le mérite d'être écrit en français !
- *Machine Learning* de Tom Mitchell est une référence très classique, écrite par un excellent spécialiste du domaine [Mit97] .
- *The elements of statistical Learning* de Hastie, Tibshirani et Friedman est un excellent livre, plus difficile que les précédents ; un bon livre de chevet pour un chercheur du domaine [HTF01].
- *Data Mining : Practical Machine Learning Tools and Techniques with Java Implementations*, de Witten et Frank. Un livre simple et pédagogique, par les auteurs de Weka (http ://www.cs.waikato.ac.nz/˜ ml/), une plate-forme écrite en java qui héberge une grande quantité d'algorithmes utilisés en apprentissage automatique et en fouille de données [WF00].

1.2 Modélisation

Dans tous les problèmes énumérés précédemment, les données observées peuvent être décrites par des mots sur un alphabet déterminé et des valeurs numériques ou symboliques. On notera X l'espace de ces données et on supposera que X peut se décomposer en un produit cartésien $X = X_1 \times \ldots \times X_n$ où chaque X_i, domaine d'un attribut A_i, est fini, égal à $\mathbb{R}$ ou à Σ^* pour un certain alphabet Σ fini.



On supposera également que les observations sont les réalisations d'une variable aléatoire X, fixe mais inconnue. On notera P la loi de distribution de X : $Pr(X = x) = P(x)$.

1.2.1 Classification supervisée

On suppose donné un ensemble fini de classes Y et pour tout élément x de X, une distribution discrète $P(\cdot|x)$ définie sur Y. Un *exemple* est un couple $(x, y) \in X \times Y$ tiré selon la distribution jointe $P(x, y) = P(x)P(y|x)$. On peut voir y comme la classe associée par le superviseur à la donnée x.

Un *échantillon* S est un ensemble fini d'exemples $\{(x_1, y_1), \ldots, (x_l, y_l)\}$ tirés indépendamment selon la distribution précédente. On parle d'exemples *i.i.d.* pour *identiquement et indépendamment distribués*.

Le problème de l'apprentissage revient alors à déterminer une fonction, appelée aussi *règle de classification* ou *classifieur*, définie de X à valeurs dans Y qui approche au mieux la réponse du superviseur.

Exemple 1. Une banque souhaite proposer une offre commerciale à certains de ses clients : une carte permettant de régler de manière sécurisée des achats sur Internet. Plutôt que d'envoyer l'offre à la totalité de ses clients, la banque préfererait cibler ceux qui sont le plus susceptibles d'êtres intéressés. Lorsqu'elle demande à ses clients d'indiquer leur coordonnées, certains indiquent spontanément une adresse e-mail : c'est peut-être un critère sur lequel baser le mailing. Un sondage réalisé sur un échantillon supposé représentatif de sa clientèle, indique que

- 40% des clients sont intéressés, 80% d'entre eux ayant indiqué leur e-mail,
- 60% ne sont pas intéressés, 40% d'entre eux ayant indiqué leur e-mail.

Cette étude incite à considérer le modèle suivant :

$X = \{email, \overline{email}\}$, $Y = \{interesse, \overline{interesse}\}$, $P(email) = 0.8 \times 0.4 + 0.6 \times 0.4 = 0.56$, $P(interesse|email) = 4/7$ et $P(interesse|\overline{email}) = 8/44 = 2/11$.

Aurait-on raison de n'envoyer l'offre qu'aux clients ayant indiqué leur e-mail ?

Remarques

1. Le modèle est *non déterministe* (puisqu'à une même description peut correspondre plusieurs classes). Le non déterminisme peut survenir dans essentiellement trois cas :
   - le problème cible est réellement non déterministe (c'est par exemple le cas lorsqu'on souhaite associer des thèmes à des textes, des articles de journaux : un article peut traiter de plusieurs sujets) ;
   - le problème est bruité ;
   - l'espace de descriptions ne décrit qu'incomplètement une situation complexe.
2. Le problème est non déterministe mais on en cherche une solution déterministe (voir exercice 1).
3. Le modèle est *non paramétrique* en ce qu'il ne présuppose aucun modèle spécifique de génération de données et qu'il n'induit aucune contrainte sur l'ensemble des fonctions que l'on doit considérer. De fait, les ensembles de fonctions que nous considérerons (réseaux de neurones, arbres de décision, ...) ne peuvent pas être assimilés aux ensembles utilisés en statistiques paramétriques : le nombre des paramètres qui

définissent chaque fonction est gigantesque et la manière dont les fonctions dépendent de ces paramètres est très complexe.

On introduit alors une *fonction de perte* (*loss function*)

$$L(y, f(x)) = \begin{cases} 0 \text{ si } y = f(x) \\ 1 \text{ sinon.} \end{cases}$$

qui mesure la différence entre la réponse y fournie par le superviseur et celle $f(x)$ fournie par la machine d'apprentissage. L'espérance mathématique de cette mesure définit la fonction *risque* (ou *erreur*)

$$R(f) = \int L(y, f(x)) dP(x, y) = \int_{y \neq f(x)} dP(x, y) = P(y \neq f(x)).$$

Le risque d'une fonction f n'est rien d'autre que la probabilité que le classifieur f prédise une réponse différente de celle du superviseur.

Le problème général de la classification supervisée peut alors s'exprimer de la manière suivante :

étant donné un échantillon $S = \{(x_1, y_1), \ldots, (x_l, y_l)\}$*, trouver une fonction* f *qui minimise le risque* $R(f)$*.*

Quelques règles de classification :

- La *règle majoritaire* : elle consiste, pour toute nouvelle instance, à retourner la classe y_{maj} majoritaire, c'est-à-dire pour laquelle $P(y)$[3] est maximum : pour tout $x \in X$,

  $$f_{maj}(x) = ArgMax_y P(y) = y_{maj} \text{ et } R(f_{maj}) = 1 - P(y_{maj}).$$

  La règle majoritaire est la plus simple. Elle peut facilement être estimée à partir de l'échantillon en calculant la classe majoritairement observée. Son erreur correspond à une valeur plancher que toute fonction de classification plus sophistiquée se doit de dépasser !
- La *règle du maximum de vaisemblance* (*maximum likelihood*) : elle consiste à retourner pour chaque instance x la classe y pour laquelle x est la valeur la plus observée.

  $$f_{mv}(x) = ArgMax_y P(x|y).$$

  Fréquemment utilisée, elle ne peut être directement estimée à partir de l'échantillon sans connaissances ou hypothèses supplémentaires.
- La *règle de Bayes* : elle consiste à retourner pour chaque instance x, la classe y dont l'observation est la plus probable, ayant observé x.

  $$f_B(x) = ArgMax_y P(y|x).$$

  Nous montrons ci-dessous que la règle de classification de Bayes est la meilleure règle de classification. Si elle pouvait être simplement estimée à partir de l'échantillon, le chapitre sur la classification s'arrêterait ici ! Malheureusement ce n'est pas le cas.

[3]Remarquer que $P(y) = \int_X P(y|x) dP(x) = \sum_{x \in X} P(y|x) P(x)$ dans le cas fini.

Proposition 1. *La règle de décision de Bayes est la règle de risque minimal.*

Démonstration. Je fais la démonstration dans le cas où les probabilités sont discrètes. La démonstration générale n'est pas plus difficile. Soit f une règle de décision. On a

$$R(f) = \sum_{f(x)\neq y} P(x,y) = \sum_{x\in X} P(x) \sum_{y\neq f(x)} P(y|x) = \sum_{x\in X} P(x)(1 - P(f(x)|x)).$$

On en déduit en particulier que

$$R(f_B) = \sum_{x\in X} P(x)(1 - P(f_B(x)|x)).$$

On sait que $P(f_B(x)|x) \geq P(y|x)$ pour tout $y \in Y$. En particulier, $P(f_B(x)|x) \geq P(f(x)|x)$. On en déduit que

$$R(f) = \sum_{x\in X} P(x)(1 - P(f(x)|x)) \geq \sum_{x\in X} P(x)(1 - P(f_B(x)|x)) = R(f_B).$$

□

Le risque du classifieur de Bayes (ou risque de Bayes) constitue donc l'erreur minimale commise par un classifieur. Il n'est nul que pour des problèmes déterministes.

Exemple 1 (suite). La majorité des clients n'étant pas intéressée par l'offre, la règle majoritaire retourne la classe $\overline{interesse}$ pour toute instance. Le risque de ce classifieur est donc de 0.4.

Comme $P(email|interesse) = 0.8 > P(email|\overline{interesse})$, la règle du maximum de vraisemblance conduira à supposer qu'un client ayant indiqué son e-mail est intéressé. Et comme $P(\overline{email}|interesse) = 1 - P(email|interesse) < P(\overline{email}|\overline{interesse})$, la règle supposera qu'un client n'yant pas indiqué son email n'est pas intéressé. Le risque de ce classifieur est : $0.4 \times 0.2 + 0.6 \times 0.4 = 0.24$.

$P(interesse|email) = 32/56 > 1/2$ et $P(interesse|\overline{email}) = 8/44$: la règle de Bayes conduit donc au même classifieur que la règle du maximum de vraisemblance.

Remarques

- On peut aussi utiliser des fonctions de perte introduisant des coûts non uniformes de mauvaise classification : il n'est pas équivalent de penser qu'un champignon vénéneux est comestible ou qu'un champignon comestible est vénéneux. On modélise cette situation en introduisant des paramètres *coût(c,c')* indiquant le coût d'un classement dans la classe c' d'une instance appartenant réellement à la classe c.
- pour des problèmes de classification binaire, où $Y = \{-1, 1\}$, on peut supposer que le classifieur recherché prendra ses valeurs dans $\mathbb{R}$, les valeurs positives (resp. négatives) de $f(x)$ correspondant à la classe 1 (resp. -1), tandis que $|f(x)|$ indique un niveau de confiance dans la classification. On peut alors considérer la fonction de perte *logistique*

$$L(y, f(x)) = \ln 1 + exp(-yf(x)).$$

1.2.2 Régression

Dans un problème de régression, y prend des valeurs continues et l'on cherche également à exprimer par une fonction la dépendance entre x et y. La fonction de perte qu'on considère principalement est l'écart quadratique défini par

$$L(y, f(x)) = (y - f(x))^2.$$

Le risque ou l'erreur d'une fonction f est alors l'écart quadratique moyen défini par :

$$R(f) = \int_{X \times Y} (y - f(x))^2 dP(x, y).$$

Comme en classification, on peut exprimer simplement une fonction qui minimise l'erreur quadratique moyen.

Proposition 2. *La fonction $\overline{f}$ définie par*

$$\overline{f}(x) = \int_Y y dP(y|x)$$

est la fonction de régression de risque minimal.

Démonstration. Soit f une fonction de régression. On a

$$\begin{aligned} R(f) &= \int_{X \times Y} (y - f(x))^2 dP(x, y) \\ &= \int_{X \times Y} (y - \overline{f}(x) + \overline{f}(x) - f(x))^2 dP(x, y) \\ &= R(\overline{f}) + \int_{X \times Y} (\overline{f}(x) - f(x))^2 dP(x, y) + 2 \int_X [(\overline{f}(x) - f(x)) \int_Y (y - \overline{f}(x)) dP(y|x)] dP(x) \end{aligned}$$

Il suffit alors de remarquer que le second terme est positif ou nul et que le troisième est nul par définition de $\overline{f}$: $\int_Y (y - \overline{f}(x)) dP(y|x) = \int_Y (y - y) dP(y|x) = 0$.

□

Remarque. La fonction $\overline{f}$ a une interprétation simple : elle calcule pour chaque élément x la moyenne des valeurs observées en x.

1.2.3 Estimation de densité

Dans les problèmes d'estimation de densité, on suppose qu'on dispose uniquement de réalisations indépendantes $x_1, \ldots, x_l$ de X et l'on cherche à estimer $P(x)$ pour tout x. On cherche donc une fonction $f : X \to [0, 1]$ qui approche P (ou sa densité) au mieux. On peut considérer la fonction de perte $L(x, y) = -\log y$ et le risque associé

$$R(f) = \int_X -\log f(x) dP(x).$$

On peut montrer que P, ou la densité de P dans le cas continu, minimise $R(f)$.

Proposition 3. *$R(f)$ est minimal*
- *lorsque $f = P$, dans le cas d'une distribution discrète,*
- *lorsque f est la densité de P, dans le cas d'une distribution continue.*

Démonstration. Dans le cas discret uniquement. Soit P' une distribution de probabilités définie sur X. On a

$$\sum_{x \in X} -P(x) \log P'(x) = \sum_{x \in X} -P(x) \log \frac{P'(x)}{P(x)} - \sum_{x \in X} P(x) \log P(x).$$

Le premier terme est la divergence de Kulback-Leibler $d_{KL}(P, P')$ de P et P'. Il suffit de montrer qu'elle est positive ou nulle pour toute distribution P' pour prouver la proposition.

Par convexité, on a

$$d_{KL}(P, P') = -\sum_{x \in X} P(x) \log \frac{P'(x)}{P(x)} \geq -\log \left(\sum_{x \in X} P(x) \frac{P'(x)}{P(x)} \right) = 0.$$

□

1.2.4 L'apprentissage en pratique

On dispose donc d'un échantillon qu'on suppose i.i.d. et l'on cherche une fonction de classification, de régression ou de densité dont le risque soit le plus faible possible. Dans la pratique, on cherche une solution dans un ensemble de fonctions $\mathcal{F}$ particulier, par exemple pouvant être calculées par des *arbres de décision*, des *réseaux de neurones*, des *fonctions linéaires*, par la méthode des k plus proches voisins, des *modèles de Markov cachés*, etc. Soit f_{opt} une fonction de risque minimal dans $\mathcal{F}$, soit f_{min} une fonction de risque minimal (on a vu plus haut que sous des hypothèses très faibles, une telle fonction existe toujours). On a bien entendu

$$R(f_{min}) \leq R(f_{opt}).$$

L'ensemble $\mathcal{F}$ doit avoir deux qualités :

1. contenir des fonctions dont le risque n'est pas trop éloigné du risque minimal $R(f_{min})$
2. permettre d'approcher un classifieur de risque minimal f_{opt} au moyen des informations dont on dispose.

1.3 Le principe de minimisation du risque empirique

Une idée naturelle pour sélectionner une fonction dans $\mathcal{F}$ est d'en chercher une qui décrit au mieux les données de l'échantillon d'apprentissage.

Le *risque empirique* $R_{emp}(f)$ d'une fonction f sur l'échantillon $S = \{(x_1, y_1), \ldots (x_l, y_l)\}$ est la moyenne de la fonction de perte calculée sur S :

$$R_{emp}(f) = \frac{1}{l} \sum_{i=1}^{l} L(y_i, f(x_i)).$$

Lorsque f est un classifieur et L est la fonction 0-1, $R_{emp}(f)$ est la moyenne du nombre d'erreurs de prédiction de f sur les éléments de S :

$$R_{emp}(f) = \frac{Card\{i|f(x_i) \neq y_i\}}{l}.$$

Lorsque f est une fonction de régression et L la fonction de perte quadratique, $R_{emp}(f)$ est la moyenne des carrés des écarts à la moyenne de f sur S :

$$R_{emp}(f) = \frac{1}{l}\sum_{i=1}^{l}(y_i - f(x_i))^2.$$

Lorsque f est une densité et L la fonction de perte correspondante, $R_{emp}(f)$ est la moyenne l'opposé de la log-vraisemblance des éléments de S :

$$R_{emp}(f) = \frac{1}{l}\sum_{i=1}^{l} - \log f(x_i).$$

Dans chacun des cas, c'est une estimation du risque réel $R(f)$ de f.

Le *principe inductif de minimisation du risque empirique (ERM)* recommande de trouver une fonction $f \in \mathcal{F}$ qui minimise $R_{emp}(f)$.

En classification, cela revient à minimiser le nombre d'erreurs commises par f sur l'échantillon ; en régression, on retrouve la méthode des moindres carrés ; en estimation de densité, on retrouve la *méthode du maximum de vraisemblance*.

Supposons que l'on soit capable de calculer dans $\mathcal{F}$ une fonction qui minimise le risque empirique et soit f_{emp} une telle fonction. On peut représenter le risque réel de la fonction f_{emp} par une somme de trois termes positifs :

$$R(f_{emp}) = R(f_{min}) + [R(f_{opt}) - R(f_{min})] + [R(f_{emp}) - R(f_{opt})]$$

Le premier est incompressible, il ne dépend que du problème considéré et donne une mesure de sa difficulté intrinsèque et au moins dans le cas de la classification et de la régression, du volume d'aléatoire, de bruit qu'il contient.

Le second terme mesure l'adéquation de l'ensemble de fonctions $\mathcal{F}$ au problème considéré. Par exemple, des discriminants linéaires peuvent ne pas être adaptés au problème à résoudre. Nous verrons que des techniques de plongements mises en œuvre avec les machines à vecteurs supports (SVM) et autres méthodes à noyaux (*kernel methods*) permettent de contourner ce problème de manque d'expressivité ; en revanche, on montre simplement que toute fonction booléenne peut être codée par un perceptron (réseau de neurones) discret à une couche cachée ; un résultat similaire peut être obtenu dans le cas continu [Cyb89]).

Le troisième représente l'erreur liée au principe de minimisation du risque empirique. On dit que le principe ERM est *consistant* dans la classe $\mathcal{F}$ si le troisième terme tend vers 0 lorsque le nombre d'exemples tend vers l'infini : dans ce cas, appliquer le principe de minimisation du risque empirique, c'est faire (asymptotiquement) au mieux. Dans les

problèmes qui se posent réellement, le nombre d'exemples disponibles est fini, limité et l'une des principales difficultés liées au principe réside dans le choix de $\mathcal{F}$.

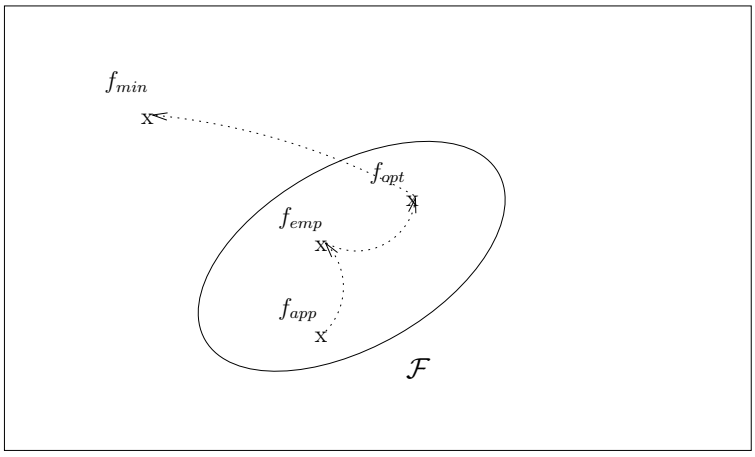
Considérons un exemple d'estimation de régression. On dispose de l couples $(x_i, y_i) \in \mathbb{R}^2$ et l'on souhaite chercher une fonction de régression polynomiale. On sait que toute fonction continue sur un intervalle fermé peut être approchée uniformément par des fonctions polynômes. Doit-on prendre pour ensemble $\mathcal{F}$ l'ensemble $\mathcal{P}$ de tous les polynômes d'une variable ou ne considérer que les polynômes de degré inférieur à d ? Et dans ce dernier cas, quelle valeur de d choisir ? En général, le principe ERM n'est pas consistant avec le premier choix : à condition que les x_i soient tous distincts, pour tout échantillon S, on pourra toujours trouver un polynôme P_S d'erreur empirique nulle mais en général, il n'y a aucune chance que ces polynômes convergent vers une solution optimale. En revanche, on peut montrer que le principe est consistant si l'on borne le degré des polynômes de $\mathcal{F}$. Il reste à trouver la meilleure valeur de d : nous verrons par la suite quelques considérations théoriques et méthodes empiriques permettant d'optimiser ce choix.

D'une manière générale, en classification, si l'espace $\mathcal{F}$ permet de mémoriser l'échantillon de travail, on obtiendra aisément une fonction de risque empirique minimum mais qui n'aura aucune capacité de généralisation (apprentissage *par cœur*).

Il faut également remarquer que dans la pratique, on ne peut pas espérer calculer une fonction qui minimise le risque empirique en un temps raisonnable : trouver la meilleure fonction compatible avec un échantillon est souvent un problème dur (au sens de la théorie de la complexité). Même pour les espaces fonctionnels les plus simples, la minimisation du risque empirique peut être une tache difficile. Considérons l'espace fonctionnel composé des fonctions de la forme

$$sgn(w \cdot x + b) = \begin{cases} 1 & \text{si } w \cdot x + b \geq 0 \\ -1 & \text{sinon,} \end{cases}$$

prenant leurs valeurs dans $\{-1, 1\}$. La frontière de décision de ces fonctions est un hyperplan de $\mathbb{R}^n$. Si l'échantillon S est exprimable par un élément de cet espace, un algorithme de programmation linéaire polynomial en trouvera un. En revanche, si S n'est pas séparable par un hyperplan, en trouver un qui fait le minimum d'erreur est un problème NP-difficile [JP78] et le problème est également difficile à approximer [HSH95], [AK94].



Il y a donc au moins quatre raisons pour lesquelles une méthode d'apprentissage appliquée à un problème particulier peut ne pas donner de résultats satisfaisants :

- la *nature non déterministe du problème*,
- la trop *faible expressivité* de l'espace fonctionnel $\mathcal{F}$ choisi,
- la *non consistance du principe ERM* ou plus généralement, du principe choisi pour approcher une fonction optimale dans $\mathcal{F}$,
- la *difficulté à minimiser le risque empirique* (ou plus généralement, à mettre en application le principe choisi).

1.4 Exercices

1. On sait qu'une pièce de monnaie est biaisée : $P(Pile) = 1/3$ et $P(Face) = 2/3$. A t-on intérêt à prédire qu'elle tombera toujours sur $Face$ ou plutôt à prédire une fois sur 3 qu'elle tombera sur $Pile$ et deux fois sur trois, qu'elle tombera sur $Face$?
2. Un cas où la règle de décision de Bayes est calculable. Un sac contient une pièce équilibrée et deux pièces biaisées : $P(Pile) = 1/3$ et $P(Face) = 2/3$.
   (a) On prélève une pièce, on la lance une fois. Quelle est la probabilité d'observer Pile ?
   (b) On prélève une pièce, on la lance deux fois. Que doit-on décider si l'on observe $Pile$ puis $Face$? La pièce est-elle biaisée ou non ?
   (c) On prélève une pièce, on la lance deux fois, on observe $Pile, Pile$. On la lance une troisième fois. Quelle est la probabilité d'observer $Pile$?
   (d) Une règle de décision doit associer une hypothèse (*biaisée, équilibrée*) à une description $\{PP, PF, FP, FF\}$. Calculez les réponses données par la règle majoritaire, la règle du maximum de vraisemblance et la règle de Bayes. Quelle est la meilleure règle de décision ?
3. La règle de Bayes peut-elle définir le même classifieur que la règle majoritaire ?

4. Une entreprise de téléphonie a constaté que ses clients se répartissaient selon deux groupes : ceux qui téléphonent en moyenne aussi souvent vers un fixe que vers un portable et ceux qui téléphonent en moyenne deux fois plus souvent vers un portable que vers un fixe. Les clients du premier groupe sont environ deux fois plus nombreux que les seconds. L'entreprise souhaite adresser une offre promotionnelle aux clients du second groupe, pour éviter qu'ils ne changent de compagnie et ce, le plus tôt possible. Comme information sur sa clientèle, l'entreprise dispose des types d'appels effectués. On notera P (resp. F) un appel vers un portable (resp. vers un fixe).

   (a) Peut-on décider sur la base des k premiers appels si un client appartient au premier groupe ou au second ? Supposons que $k = 4$ et qu'on observe la suite P, P, F, P. Quelle décision conduit à prendre la règle majoritaire ? la règle du maximum de vraisemblance ? la règle de Bayes ?

   (b) Supposons que le bénéfice de l'opération soit mesuré selon la matrice suivante :

Prédit/Réel	1	2
1	0	0
2	-1	2

   Quelle décision l'observation de la suite P, P, F, P doit-elle entraîner ?

5. **Le classifieur naïf de Bayes.** Le tableau ci-dessous récapitule les conditions qui ont accompagné les succès et les échecs d'une équipe de football. Est-il possible de prédire l'issue d'un match en fonction des conditions dans lesquelles il se déroule ?

Match à domicile ?	Balance positive ?	Mauvaises conditions climatiques ?	Match précédent gagné ?	Match gagné
V	V	F	F	V
F	F	V	V	V
V	V	V	F	V
V	V	F	V	V
F	V	V	V	F
F	F	V	F	F
V	F	F	V	F

FIG. 1 – Jeu de données *FootBall*.

Les conditions d'un match sont modélisées par un élément $\mathbf{x}$ de $X = \{V, F\}^4$, correspondant aux valeurs des attributs figurant sur la première ligne du tableau. D'après la règle de classification de Bayes, il suffit de connaître $P(V|\mathbf{x})$ pour pouvoir classer $\mathbf{x}$ de manière optimale : $f(\mathbf{x}) = V$ si $P(V|\mathbf{x}) \geq 1/2$ et $f(\mathbf{x}) = F$ sinon.

D'après la formule de Bayes, on a :

$$P(V|\mathbf{x}) = \frac{P(\mathbf{x}|V)P(V)}{P(\mathbf{x})} \text{ et } P(F|\mathbf{x}) = \frac{P(\mathbf{x}|F)P(F)}{P(\mathbf{x})}$$

soit encore

$$P(V|\mathbf{x}) \geq 1/2 \text{ ssi } P(\mathbf{x}|V)P(V) \geq P(\mathbf{x}|F)P(F).$$

On peut évaluer $P(V)$ et $P(F)$ en comptant le nombre de matchs gagnés et perdus :

$$\hat{P}(V) = 4/7 \text{ et } P(F) = 3/7.$$

L'évaluation de $P(\mathbf{x}|V)$ et de $P(\mathbf{x}|F)$ est plus délicate. La règle *naïve* de Bayes consiste à faire l'hypothèse que les attributs décrivant $\mathbf{x}$ sont indépendants conditionnellement à chaque classe : si l'on écrit $\mathbf{x} = (x_1, x_2, x_3, x_4)$, on suppose que

$$P(\mathbf{x}|V) = \prod_{i=1}^{4} P(x_i|V) \text{ et } P(\mathbf{x}|F) = \prod_{i=1}^{4} P(x_i|F).$$

Pour estimer $P(\mathbf{x}|V)$ et $P(\mathbf{x}|F)$, il suffit alors d'estimer $P(x_i = V|V)$ et $P(x_i = V|F)$ pour $i = 1, \ldots, 4$.

(a) Réaliser ces estimations

(b) Classer l'élement (V, F, V, F)

2 Les arbres de décision

2.1 Définition

Etant donnés n attributs $A_1, \ldots, A_n$, l'espace de description X est le produit cartésien des domaines X_i de chaque attribut A_i.

$$X = \prod_{i=1}^{n} X_i \text{ où } X_i = Dom(A_i).$$

Les attributs peuvent être :
– binaires,
– n-aires,
– réels.

Les *arbres de décision* sont des règles de classification qui basent leur décision sur une suite de tests associés aux attributs, les tests étant organisés de manière arborescente.

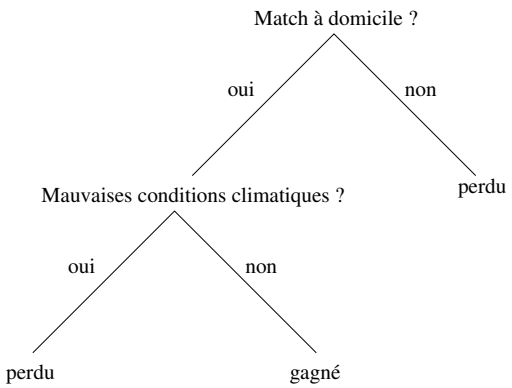


FIG. 2 – La situation en début d'un match de football est décrite par deux attributs binaires : *Match à domicile* et *Mauvaises conditions climatiques*. L'attribut classe prend deux valeurs : *gagné* et *perdu*. Un exemple d'arbre de décision

Un *arbre de décision* est un arbre au sens informatique du terme. Les nœuds internes sont appelés *nœuds de décision*. Chaque nœud de décision est étiqueté par un *test* qui peut être appliqué à toute description d'un individu de la population. En général, chaque test examine la valeur d'un unique attribut de l'espace des descriptions. Les réponses possibles au test correspondent aux étiquettes des arcs issus de ce nœud. Les *feuilles* sont étiquetées par une classe appelée *classe par défaut*. Chaque nœud interne ou feuille est repéré par sa *position* : la liste des numéros des arcs qui permettent d'y accéder depuis la racine. Les positions des nœuds de l'arbre de déicion de la figure 2 sont : $\epsilon, 1, 2, 1 \cdot 1, 1 \cdot 2$.

A tout arbre de décision, on associe de façon naturelle une procédure de classification. En effet, à toute description complète est associée une et une seule feuille de l'arbre de décision. Cette association est définie en commençant à la racine de l'arbre et en descendant

dans l'arbre selon les réponses aux tests qui étiquettent les nœuds internes. La classe associée est alors la classe par défaut associée à la feuille qui correspond à la description. Cette procédure de classification a une traduction immédiate en terme de règles de décision (une règle par branche). Les systèmes de règles qu'on obtient ainsi ont la particularité que l'ordre dans lequel on examine les attributs est fixé et que les règles de décision sont mutuellement exclusives.

Les arbres de décision ont deux qualités appréciables :
- les décisions sont aisément interprétables,
- la classification est très rapide.

2.2 Classification supervisée par arbres de décision

Les premiers algorithmes de classification par arbres de décision sont anciens [MS63]. Les deux travaux les plus marquants sont la création de CART [BFOS84] et la création de C4.5 [Qui93].

Nous allons considérer l'exemple très simple suivant pour introduire les algorithmes d'apprentissage par arbres de décision. La tableau ci-dessous récapitule les conditions qui ont accompagné les succès et les échecs d'une équipe de football. Est-il possible de prédire l'issue d'un match en fonction des conditions dans lesquelles il se déroule ?

Match à domicile ?	Balance positive ?	Mauvaises conditions climatiques ?	Match précédent gagné ?	Match gagné
V	V	F	F	V
F	F	V	V	V
V	V	V	F	V
V	V	F	V	V
F	V	V	V	F
F	F	V	F	F
V	F	F	V	F
V	F	V	F	F

FIG. 3 – Jeu de données *FootBall*.

Il n'est pas difficile de construire un arbre de décision qui fasse le moins d'erreurs possible sur ce jeu de données, c'est-à-dire de risque empirique minimal. D'une manière générale, il est facile de construire un arbre de déciosion de risque empirique minimal sur n'importe quel jeu de données. Voir exercice 1.

Mais il n'y a aucune raison qu'un tel arbre de décision ait de bonnes qualités en généralisation. Pourquoi ? Essentiellement parce qu'il y a sans doute un grand nombre d'arbres de décision différents ayant cette propriété et qu'un arbre choisi aléatoirement parmi eux n'a aucune chance de faire partie des arbres ayant les meilleures performances prédictives.

Y a-t-il des raisons de penser que *le plus petit* arbre de décision compatible avec les données aura de meilleures qualités de généralisation ? La réponse est oui. Intuitivement, c'est parce que cela revient à rechercher une solution dans un espace plus petit, dans lequel l'adéquation aux données a une forte probabilité de ne pas être fortuite. Cette intuition peut être vérifiée et quantifiée : c'est l'un des thèmes traités par la *théorie de l'apprentissage statistique*, développée par Vladimir Vapnik.

Quoiqu'il en soit, trouver le plus petit arbre de décision compatible avec un jeu de données est un problème NP-complet [BR92]. On peut néanmoins tenter de construire un *petit* arbre de décision compatible avec le maximum de données. Cela est conforme à des principes souvent invoqués en apprentissage :

- *Le principe du rasoir d'Occam* : trouver l'hypothèse la plus courte possible compatible avec les données.
- *Le principe MDL (Minimum Description Length)* : étant donner des données D, trouver l'hypothèse H telle que $|H| + |D/H|$, la longueur d'un codage des données via l'hypothèse H, soit la plus petite possible.

Ces heuristiques sont-elles pertinentes ? Qu'est-ce qui les fonde ? C'est aussi un des thèmes étudiés en théorie de l'apprentissage statistique.

2.2.1 Algorithmes d'apprentissage par arbres de décision

Les principaux algorithmes d'apprentissage par arbres de décision sont CART (Breiman, [BFOS84]), C4.5 (Quinlan, [Qui94]). Nous présentons ci-dessous quelques une de leurs caractéristiques.

La première étape consiste à contruire un petit arbre consistant avec la plupart des données.

Idée centrale : Diviser récursivement et le plus efficacement possible les exemples de l'ensemble d'apprentissage par des tests définis à l'aide des attributs jusqu'à ce que l'on obtienne des sous-ensembles d'exemples ne contenant (presque) que des exemples appartenant à une même classe. Cette idée débouche sur des méthodes de construction Top-Down, c'est-à-dire construisant l'arbre de la racine vers les feuilles, gloutonnes et récursives.

Dans toutes les méthodes, on trouve les trois opérateurs suivants :

1. **Décider si un nœud est terminal**, c'est-à-dire décider si un nœud doit être étiqueté comme une feuille ou porter un test.
2. **Si un nœud n'est pas terminal, sélectionner un test à lui associer**.
3. **Si un nœud est terminal, lui affecter une classe**.

On peut alors définir un schéma général d'algorithme, sans spécifier comment seront définis les 3 opérateurs décrits plus haut :

Algorithme d'apprentissage générique
entrée : échantillon S
début

Initialiser l'arbre courant à l'arbre vide ; la racine est le nœud courant
répéter
    Décider si le nœud courant est terminal
    Si le nœud est terminal **alors**
        Lui affecter une classe
    sinon
        Sélectionner un test et créer autant de nouveaux nœuds fils
        qu'il y a de réponses possibles au test
    FinSi
    Passer au nœud suivant non exploré s'il en existe
Jusqu'à obtenir un arbre de décision
fin

En général, on décide qu'un nœud est terminal lorsque tous les exemples correspondant à ce nœud, ou du moins la plupart d'entre eux sont dans la même classe, ou encore, s'il n'y a plus d'attributs non utilisés dans la branche correspondante, ...

En général, on attribue au nœud la classe majoritaire (éventuellement calculée à l'aide d'une fonction de coût lorsque les erreurs de prédiction ne sont pas équivalentes). Lorsque plusieurs classes sont en concurrence, on peut choisir la classe la plus représentée dans l'ensemble de l'échantillon, ou en choisir une au hasard.

La sélection d'un test à associer à un nœud est plus délicate. Puisqu'on cherche à construire un arbre de décision le plus petit possible rendant compte au mieux des données, une idée naturelle consiste à chercher un test qui fait le plus progresser dans la tache de classification des données d'apprentissage. Comment mesurer cette progression ? CART utilise l'*indice de Gini* et C4.5 utilise la notion d'*entropie*.

Soit S un échantillon et $S_1, \ldots, S_k$ la partition de S selon les classes de l'attribut cible. On définit

$$Gini(S) = \Sigma_{i=1}^{k} |S_i|/|S| \times (1 - |S_i|/|S|) = \Sigma_{i \neq j}^{c} |S_i|/|S| \times |S_j|/|S|$$
$$Ent(S) = -\Sigma_{i=1}^{k} |S_i|/|S| \times \log(|S_i|/|S|)$$

Remarquer que selon la base du logarithme utilisé, l'entropie varie d'un terme multiplicatif constant.

Considérons le cas de deux classes et appelons x la proportion d'éléments de classe 1 et donc $1 - x$ la proportion d'éléments de classe 2. On a donc

$$Gini(S) = 2x(1 - x)$$
$$Ent(S) = -x \log x - (1 - x) \log(1 - x)$$

Ces fonctions prennent leurs valeurs dans l'intervalle $[0, 1]$, sont nulles pour $x = 0$ et $x = 1$, ont leur maximum pour $x = 1/2$ ($1/2$ pour l'indice de Gini et 1 pour l'entropie calculée avec le logarithme de base 2). Ces propriétés restent vraies lorsque l'attribut cible prend k

valeurs (k classes) : ces fonctions sont positives, nulles si et seulement si il existe un indice i tel que $S = S_i$, et maximales lorsque pour tous les indices i, les S_i ont le même nombre d'éléments, $Card(S)/k$.

Considérons le cas où les attributs descriptifs sont binaires. Soit p la position courante de l'arbre en construction et T un test potentiel. On définit

$$Gain_{Gini}(p,T) = Gini(S_p) - \sum_{j=1}^{2} P_j \times Gini(S_{p_j})$$

$$Gain_{Ent}(p,T) = Ent(S_p) - \sum_{j=1}^{2} P_j \times Ent(S_{p_j})$$

où S_p est l'échantillon associé à la position p et où P_i est la proportion des éléments de S_p qui satisfont la i-ème branche du test T.

Sélectionner l'attribut dont le gain est maximum correspond à la stratégie gloutonne énoncée plus haut : rechercher le test faisant le plus progresser la classification. Le gain est maximal lorsque le choix d'un attribut permet de classer correctement toutes les données ; il est nul lorsque les données sont aussi mal classées après le test qu'avant.

Rermarquer que le premier terme du gain ne dépend pas de l'attribut T. Maximiser le gain revient donc à minimiser $\sum_{j=1}^{2} P_j \times Gini(S_{p_j})$ ou $\sum_{j=1}^{2} P_j \times Ent(S_{p_j})$ selon le critère utilisé.

Pour notre exemple courant, avec le critère de Gini et en désignant les attributs descriptifs par *Dom, Bal, MCC* et *MPG*, nous avons :

- $Gain(\epsilon, Dom) = Gini(S) - (\frac{5}{8}Gini(S_1) + \frac{3}{8}Gini(S_2)) = Gini(S) - \frac{5}{8}\frac{2}{5}\frac{3}{5} - \frac{3}{8}\frac{1}{3}\frac{2}{3} = Gini(S) - \frac{7}{30}$
- $Gain(\epsilon, Bal) = Gini(S) - \frac{3}{16}$
- $Gain(\epsilon, MCC) = Gini(S) - \frac{7}{30}$
- $Gain(\epsilon, MPG) == Gini(S) - \frac{1}{4}$

Le gain maximal est obtenu pour le choix du test *Balance positive ?*. Si l'on poursuit la construction de l'arbre de décision, on obtient l'arbre de la figure 4.

L'arbre construit est d'erreur apparente nulle mais il se peut que l'erreur réelle soit importante, c'est-à-dire que l'arbre construit soit bien adapté à l'échantillon mais ait un pouvoir de prédiction faible. La figure 5 représente un compte-rendu d'expériences : à chaque étape de construction d'un arbre de décision selon la méthode décrite précédemment, on a reporté l'erreur empirique de l'arbre courant sur l'échantillon d'apprentissage $R^S_{emp}(f)$ et l'erreur réelle $R(f)$ de cet arbre (par exemple estimée sur un échantillon test indépendant de grande taille). On constate que l'erreur apparente $R^S_{emp}(f)$ diminue constamment lors de la construction de l'arbre mais que l'erreur réelle $R(f)$ diminue, se stabilise, puis augmente. Il s'agit là d'un comportement couramment observé et bien compris : au début du processus, l'arbre apprend des caractéristiques générales de la population ; à partir d'un certain moment, l'arbre apprend des propriétés spécifiques de l'échantillon. On parle alors de *sur-apprentissage* ou d'*apprentissage par cœur*.

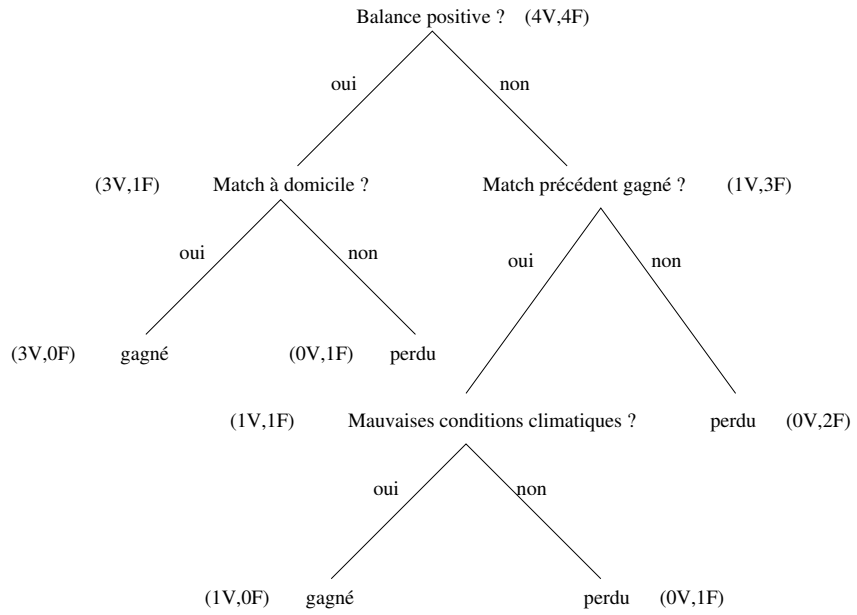


FIG. 4 – L'arbre de décision construit en utilisant le critère de Gini.

Pour éviter de construire un arbre trop grand, on peut rechercher un critère qui permette d'arrêter la croissance de l'arbre au bon moment. Une méthode consiste à réserver un *ensemble de validation* S_{val}, c'est-à-dire un ensemble d'exemples qui ne devront pas être utilisés pour construire l'arbre puis à estimer l'erreur réelle $R(f)$ de l'arbre courant par l'erreur empirique $R^{val}_{emp}(f)$ sur cet ensemble de validation. Si l'ensemble de validation n'est pas trop petit, on doit constater que cette estimation diminue, puis stagne puis augmente, au fur et à mesure de la construction de l'arbre : il suffit d'arrêter la construiction lorsque l'estimation de l'erreur réelle ne diminue plus (*early stopping*). Cette méthode n'est pas optimale car elle ne permet pas de revenir sur des mauvais choix faits avant que l'on constate que l'erreur réelle ne diminue plus.

Les méthodes les plus utilisées procèdent plutôt en deux phases. L'algorithme présenté ci-dessus est utilisé pour construire un arbre de décision sans arrêt prématuré ; dans une seconde phase, l'arbre obtenu est *élagué* de façon à faire diminuer l'erreur réelle : élaguer un arbre consiste à en supprimer certains sous-arbres.

2.2.2 Elagage d'un arbre de décision

Plusieurs méthodes d'élagage ont été étudiées, voir par exemple [EMS97]. Nous présentons la méthode utilisée par CART.

Soit T_0 un arbre de décision et $\mathcal{T}$ l'ensemble de tous les arbres de décision que l'on peut construire à partir de T_0 en remplaçant certains nœuds internes par des feuilles étiquetées comme décrit précédemment.

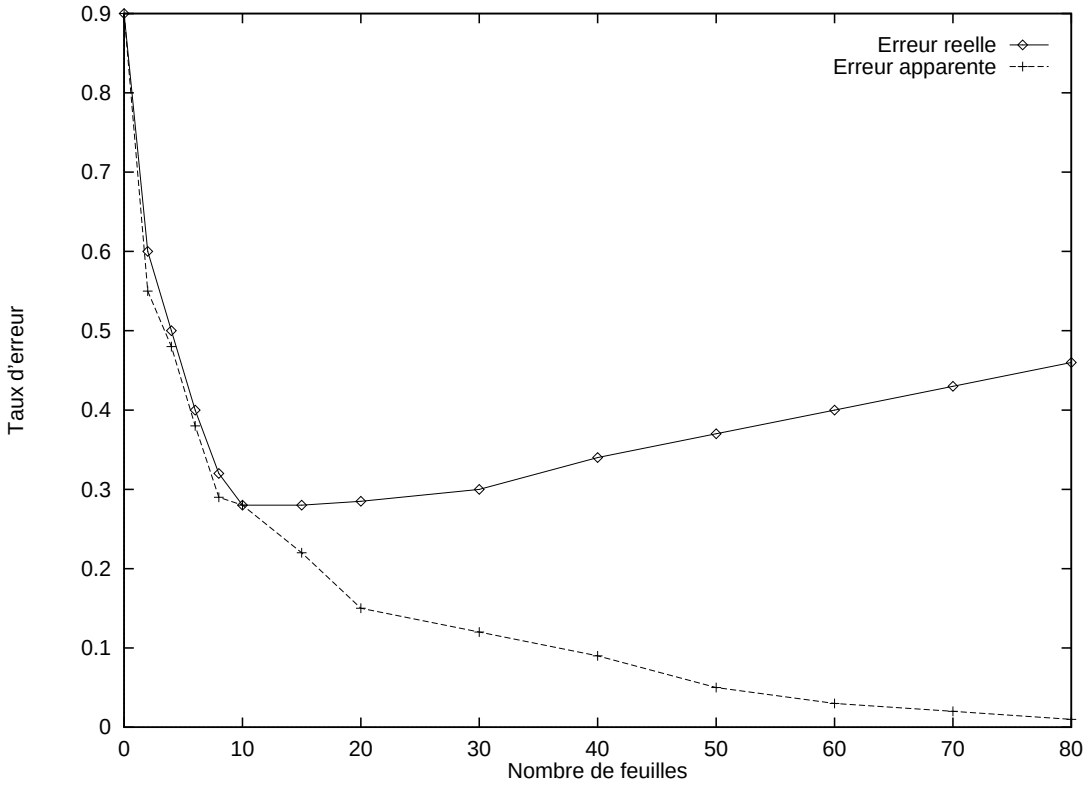


FIG. 5 – Erreur réelle et erreur apparente pour des données réelles de reconnaissance de caractères ([BFOS84])

Pour chaque nœud interne p de T_0, on définit le rapport

$$\alpha = \frac{\Delta R^S_{emp}}{|T_p| - 1}$$

où ΔR^S_{emp} est le nombre d'erreurs supplémentaires que commet l'arbre de décision sur S lorsqu'on l'élague à la position p et où $|T_p| - 1$ mesure le nombre de feuilles supprimées.

L'arbre T_{i+1} est obtenu en élaguant T_i en ses nœuds qui portent la plus petite valeur de α. On obtient ainsi une suite $T_0, \ldots, T_i, \ldots, T_t$ d'éléments de $\mathcal{T}$, dont le dernier est réduit à une feuille.

Dans une deuxième phase, on cherche la valeur optimale de α. On peut procéder en utilisant un ensemble de validation ou par *validation croisée* (voir [EMS97]).

Dans le premier cas, soit S_{val} un ensemble d'exemples qui n'ont pas été utilisés pour construire T_0 et qui ont été générés dans les mêmes conditions que S. Choisir l'arbre de la suite $T_0, T_1, \ldots, T_t$ dont le nombre d'erreurs sur S_{val} est minimal.

Exemple. Reprenons l'exemple précédent. Supposons que l'on ait réservé l'ensemble de validation suivant :

Match à domicile ?	Balance positive ?	Mauvaises conditions climatiques ?	Match précédent gagné ?	Match gagné
V	V	V	F	V
F	V	V	F	V
F	F	F	V	F
V	F	V	F	F
V	F	V	F	V

L'arbre t_0 est celui de la figure 4. Pour cet arbre, on a : $g(\epsilon) = 4/4 = 1, g(1) = 1/2, g(2) = 1/3, g(2 \cdot 1) = 1/2$. L'arbre t_1 est donc obtenu à partir de t_0 en élaguant la branche à partir de le position 2 (voir figure 6). Pour l'arbre t_1, on a $g(\epsilon) = 3/3 = 1, g(1) = 1/2$. L'arbre t_2 est donc obtenu à partir de t_1 en élaguant la branche à partir de le position 1. L'arbre t_3 est l'arbre réduit à une feuille, portant par exemple la classe *gagné*. Sur l'ensemble de validation, les arbres t_0, t_1 et t_2 commetent une erreur et l'arbre t_3 en commet 2. L'algorithme d'élagage retournera alors l'arbre t_2.

Remarque. C4.5 n'utilise pas d'ensemble de validation mais essaie plutôt d'estimer l'erreur réelle à partir de l'erreur apparente en la corrigeant par un a priori pessimiste (voir exercice 4).

2.3 Compléments

Nous avons supposé dans ce qui précède que les attributs descriptifs étaient binaires : il n'est pas très difficile d'étendre les techniques précédentes aux attributs n-aires et continus.

Attributs continus

On associe à un attribut continu A des tests binaires de la forme $A < a$ ou $A \leq a$. Si les valeuirs prises par A dans l'échantillo d'apprentissage sont $a_1 < a_2 < \ldots < a_N$, on envisagera les tests

$$A < \frac{a_i + a_{i+1}}{2}$$

pour $i = 1$ à $N - 1$. Les algorithmes décrits dans les sections précédentes peuvent être alors appliqués tels quels.

Attributs n-aires

On peut envisager plusieurs méthodes pour tenir compte des attributs n-aires.

1. Si l'attribut cible est binaire et prend par exemple ses valeurs dans $\{0, 1\}$, on peut classer les valeurs $a_1, \ldots, a_N$ prises par l'attribut A selon le nombre d'éléments de l'échantillon S classés 1 : $a_i < a_j$ ssi $Card(\{(x, 1) \in S | A(x) = a_i\}) < Card(\{(x, 1) \in S | A(x) = a_j\})$. On procède alors comme dans le cas continu.

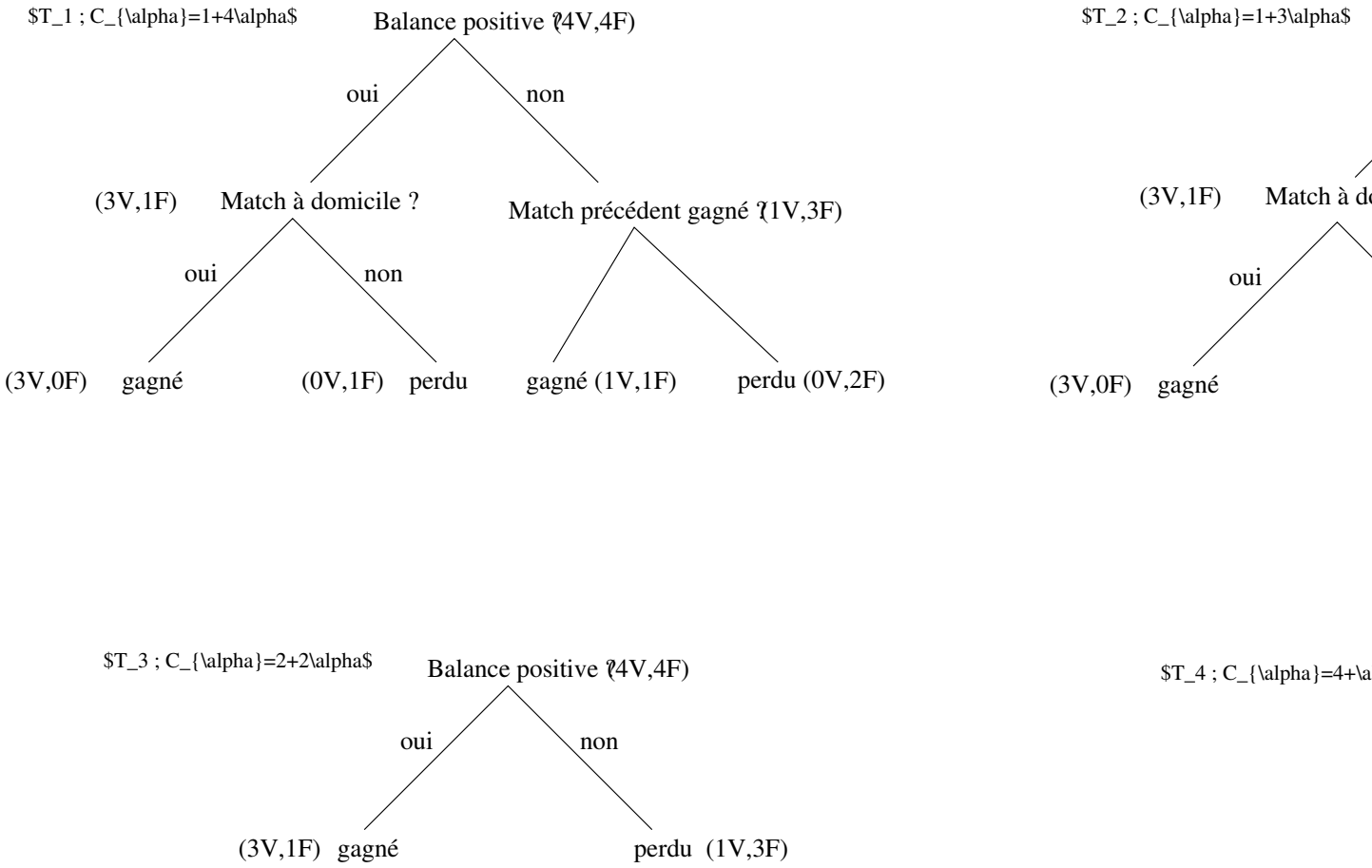


FIG. 6 – De haut en bas, la suite des arbres élagués : t_1, t_2 et t_3.

2. On peut également prolonger directement les formules de gain définies pour un attribut binaire.

$$Gain_{Gini}(p,T) = Gini(S_p) - \sum_{j=1}^{N} P_j \times Gini(S_{p_j})$$

$$Gain_{Ent}(p,T) = Ent(S_p) - \sum_{j=1}^{N} P_j \times Ent(S_{p_j})$$

   L'inconvénient est qu'on privilégie ainsi les attributs ayant une grande arité. Pour s'en convaincre, considérer le cas où l'attribut est une clé identifiant chaque élément de l'échantillon : un test portant sur cet attribut aura nécessairement un gain maximum.

3. Pour remédier au problème mentionné ci-dessus, on peut introduire un terme pénalisant les attributs de grande arité. Dans C4.5, l'algorithme utilise la notion de *GainRatio* à la place du gain : un attribut A d'arité N et prenant les valeurs $a_1, \ldots, a_N$ sera

évalué par

$$GainRatio = \frac{Gain}{-\sum_{i=1}^{N} \frac{k_i}{S} \log \frac{k_i}{S}}$$

où S est l'échantillon courant et où k_i est le nombre d'élément de S pour lesquels A prend la valeur a_i.

Quelques développements complémentaires :

- On peut modifier les algorithmes précédents de manière à pouvoir prendre en compte une matrice de coûts de prédictions erronées : il n'est pas équivalent de prédire qu'un champigno comestible est vénéneux ou le contraire !
- Dans de nombreuses situations, les données dont on dispose sont incomplètement renseignées : comment tenir compte des valeurs manquantes. Voir l'exercice 3.
- Des attributs n-aires peuvent prendre un grand nombre de valeurs que l'on peut souhaiter regrouper : si le but de l'apprentissage est d'étudier la corrélation entre la couleur d'une voiture et la fréquence des accidents de son propriétaire, faut-il distinguer les 20 nuances de rouge proposées par le constructeur ?
- Les arbres de décision peuvent être utilisés en régression : les techniques d'apprentissage ne sont pas très éloignées de celles que nous avons vu en classification. Voir par exemple [HTF01].
- Un des inconvénients principaux des méthodes d'apprentissage par arbres de décision est leur *instabilité*. Sur des données réelles, il s'en faut souvent de peu qu'un attribut soit choisi plutôt qu'un autre et le choix d'un attribut-test, surtout s'il est près de la racine, influence grandement le reste de la construction. La conséquence de cette instabilité est que les algorithmes d'apprentissage par arbres de décision ont une *variance* importante, qui nuit à la qualité de l'apprentissage. Des méthodes comme le *Bagging* (pour Bootstrap Aggregating, voir par exemple [HTF01]) ou les Random Forests [Bre01] permettent dans une certaine mesure de remédier à ce problème.

2.4 Exercices

Exercice 1. Construisez un arbre de décision de risque empirique nul sur le jeu de données de la figure 3. Décrivez dans le cas général un algorithme de construction efficace d'un arbre de décision d'erreur empirique minimal.

Exercice 2. Une banque dispose des informations suivantes sur un ensemble de clients :

client	M	A	R	E	I
1	moyen	moyen	village	oui	oui
2	élevé	moyen	bourg	non	non
3	faible	âgé	bourg	non	non
4	faible	moyen	bourg	oui	oui
5	moyen	jeune	ville	oui	oui
6	élevé	âgé	ville	oui	non
7	moyen	âgé	ville	oui	non
8	faible	moyen	village	non	non

Construisez les arbres de décision correspondant à ce jeu de données, en utilisant l'indice de Gini puis l'entropie.

Exercice 3. On considère un espace de description comprenant les trois attributs *forme*, *taille* et *couleur* prenant respectivement les valeurs *rond* et *carré*, *petit* et *grand*, *bleu*, *blanc* et *rouge*. L'attribut cible est binaire de valeurs *oui* et *non*.

Les données disponibles sont les suivantes (le ? correspond à une valeur manquante) :

forme	**taille**	**couleur**	**classe**
rond	petit	bleu	oui
carré	grand	rouge	non
rond	?	blanc	oui
carré	petit	bleu	oui
rond	grand	bleu	oui
carré	grand	blanc	non
carré	?	blanc	oui
carré	grand	bleu	non
carré	petit	rouge	oui
rond	grand	blanc	oui

Valeur majoritaire de l'attribut On remplace les valeurs manquantes par la valeur majoritaire prise par cet attribut sur l'échantillon complet. Quelle valeur associe-t-on sur notre échantillon ? Peut-on trouver un arbre de décision parfait ? Appliquer l'algorithme de construction d'arbre de décision en utilisant l'entropie pour le calcul du gain. On décide qu'un nœud est terminal, i.e. d'attribuer une feuille, lorsqu'il y a au plus un exemple mal classé associé à ce nœud. Vous détaillerez les calculs pour le test à choisir en racine de l'arbre.

Valeur majoritaire de l'attribut par classe Étant donné un exemple avec une valeur manquante, on remplace la valeur manquante par la valeur majoritaire prise par l'attribut correspondant pour les exemples de l'échantillon appartenant à la même classe. Quelles valeurs associe-t-on sur notre échantillon ? Peut-on trouver un arbre de décision parfait ? Quel arbre obtient-on en appliquant l'algorithme basé sur l'entropie ?

Méthode utilisée par C45 Cette méthode consiste à ne plus attribuer une valeur à l'attribut mais une probabilité pour chacune des valeurs possibles. Ces probabilités sont estimées par les fréquences des valeurs possibles de cet attribut pour l'échantillon associé à une position p de l'arbre en construction. Par exemple, à la racine, la probabilité que l'attribut taille ait la valeur petit est de 3/8 car il y a 8 exemples pour lesquels la valeur de l'attibut taille est connue et 3 ont la valeur petit. Quelles seraient les modifications à apporter à l'algorithme ?

Exercice 4. Méthode d'élagage de C4.5 La méthode classique d'apprentissage par arbres de décision consiste à construire un arbre à l'aide d'un échantillon d'apprentissage puis à l'élaguer à l'aide d'un échantillon test. L'idée sous-jacente est que la phase d'élagage doit permettre d'améliorer l'erreur réelle et que seul l'échantillon test permet de l'estimer de manière à peu près fiable.

Mais cette idée ne vaut que si l'on dispose de suffisamment d'exemples pour la première phase : un arbre dont l'erreur est catastrophique ne donnera jamais de bons résultats, quelque soit l'élagage qu'on lui fera subir !

Une idée récurrente consiste à apprendre avec tous les exemples disponibles et à élaguer avec ces mêmes exemples. Mais cette idée ne peut fonctionner qu'à condition de ne pas se baser sur l'erreur apparente calculée sur l'ensemble d'apprentissage pour estimer l'erreur réelle.

Quinlan propose la méthode suivante dans C4.5 :

On introduit un paramètre de confiance CF (par défaut, ce paramètre vaut 25%). Pour chaque feuille de l'arbre, notons N le nombre d'exemples qu'elle couvre et E le nombre d'erreurs de classification qu'elle induit dans l'échantillon. Soit p la probabilité pour qu'un nouvel exemple soit mal classé par cette feuille. La quantité E/N est donc un estimateur de p. Pour tenir compte du fait que l'arbre construit n'est pas indépendant des données, nous allons supposer que cet estimateur est très optimiste.

Plus précisément, soit E_p une variable aléatoire de loi binomiale de paramètres (N, p). C'est-à-dire que

$$Pr(E_p = k) = C_N^k p^k (1-p)^{N-k} \text{ pour } 0 \leq k \leq N$$

Nous posons $p(E, N) = max\{p | Pr(E_p \leq E) \geq CF\}$. Nous prendrons $p(E, N)$ comme valeur estimée de l'erreur réelle pour cette feuille.

Exemple : supposons qu'une feuille couvre $N = 4$ exemples et supposons qu'elle induise une erreur ($E = 1$). On prend $CF = 25\%$. On a :

$$\begin{aligned} p(E, N) &= max\{p | Pr(E_p \leq 1) \geq 0,25\} \\ &= max\{p | Pr(E_p = 0) + Pr(E_p = 1) \geq 0,25\} \\ &= max\{p | (1-p)^4 + 4p(1-p)^3 \geq 0,25\} \end{aligned}$$

Pour cet exemple, on trouve $p \simeq 0,54$. Autrement dit, on estime par ce procédé l'erreur réelle pour cette feuille à 54% (au lieu des 25% fournis par l'erreur apparente).

Le reste est plus classique : on calcule l'erreur réelle estimée d'un arbre en faisant une somme pondérée des erreurs réelles estimées de ses fils.

Par exemple, si un nœud A a trois fils $A1$, $A2$ et $A3$, si le nombre d'exemples couverts par chacun de ces fils est respectivement de $N1$, $N2$ et $N3$, et si les erreurs réelles estimées pour chacun de ces fils sont $e1$, $e2$ et $e3$ alors l'erreur réelle estimée de A sera de $(N1e1 + N2e2 + N3e3)/(N1 + N2 + N3)$.

Pour élaguer un arbre, on applique l'algorithme suivant :

Tant qu'il existe un sous-arbre que l'on peut remplacer par une feuille sans faire croître l'erreur réelle estimée **alors** élaguer ce sous-arbre.

Application : On considère un espace de description comprenant deux attributs *adoption* et *education* pouvant prendre chacun trois valeurs : y, n et u. On suppose que l'attribut cible est binaire et que ses valeurs sont A et B.

On considère l'arbre suivant :

```
adoption = y : A (0;151)
adoption = u : A (0;1)
adoption = n :
   education = n : A (0;6)
   education = y : A (0;9)
   education = u : B (0;1)
```

Chacun des couples (0 ; 151), ..., est de la forme $(E; N)$.

On donne $p(0; 6) = 0,206$; $p(0; 9) = 0,143$; $p(0; 1) = 0,750$; $p(1; 16) = 0,159$; $p(0; 151) = 0,009$; $p(1; 168) = 0,016$.

1. Calculez l'erreur réelle estimée pour l'arbre

   ```
   adoption = n :
      education = n : A (0;6)
      education = y : A (0;9)
      education = u : B (0;1)
   ```

2. Calculez l'erreur réelle estimée pour l'arbre complet
3. Peut-on élaguer le sous-arbre de racine `adoption = n`
4. Peut-on remplacer l'arbre entier par une feuille ?
5. Après élagage, quelle est l'erreur réelle estimée ?

3 Quelques notions élementaires de probabilités et statistiques

3.1 Rappels

Un *univers* Ω est un ensemble modélisant les réalisations possibles d'une expérience. On appelle *évènement* toute partie de Ω et *distribution de probabilité* toute fonction P associant un nombre réel $P(A)$ à tout événement A[4] et vérifiant les trois axiomes suivants :

- $P(A) \geq 0$ pour tout A,
- $P(\Omega) = 1$,
- pour toute collection $(A_i)_{i\in I}$ d'évènements disjoints,

$$P(\cup_{i\in I} A_i) = \sum_{i_i n I} P(A_i).$$

Lorsque l'univers Ω est fini ou dénombrable, pour tout évènement $A \subseteq \Omega$, $P(A) = \sum_{x\in A} P(x)$.

Quelques propriétés élémentaires :

- $P(\emptyset) = 0$,
- $P(A \cup B) = P(A) + P(B) - P(A \cap B)$,
- $P(\overline{A}) = 1 - P(A)$,
- $A \subseteq B \Rightarrow P(A) \leq P(B)$,
- $0 \leq P(A) \leq 1$ pour tout A.

Si B est un évènement de probabilité non nulle, on définit la distribution de probabilité $P(\cdot|B)$ sur Ω par $P(C|B) = P(C \cap B)/P(B)$. En particulier, $P(x|B) = P(x)/P(B)$ si $x \in B$ et $P(x|B) = 0$ sinon.

Pour tout évènement A, on a :

- $P(A|B) = P(A \cap B)/P(B)$,
- $P(A|B) = P(B|A)P(A)/P(B)$ (formule de Bayes).

Si $(A_i)_{i\in I}$ est une famille d'évènements deux à deux incompatibles et telle que $\cup_{i\in I} A_i = \Omega$, alors pour tout évènement B, on a

$$P(B) = \sum_{i\in I} P(B|A_i)P(A_i) \text{ (formule des probabilités totales).}$$

On dit que deux évènements A et B sont *indépendants* si $P(A \cap B) = P(A)P(B)$. Si les probabilités de A et B sont non nulles, l'indépendance de A et B est équivalente à $P(A) = P(A|B)$ ou $P(B) = P(B|A)$.

[4]Lorsque l'univers n'est pas dénombrable, il n'est pas toujours possible d'associer une probabilité à chacune de ses parties et l'on doit définir une notion plus complexe de σ-algèbre d'évènements. On ne considérera pas ce cas ici.

3.2 Variables aléatoires

Soit P une distribution de probabilité sur l'univers Ω. Une variable aléatoire réelle est une application $X : \Omega \to \mathbb{R}$.

On appelle loi de probabilité de X la donnée des probabilités $P(X \in I) = P(X^{-1}(I))$ pour tout intervalle I de $\mathbb{R}$. Si X ne prend qu'un nombre fini ou dénombrable de valeurs, la loi de probabilité de X est définie par les valeurs $P(X = \alpha)$ pour $\alpha \in X(\Omega)$.

Exemples de variables aléatoires :

1. On jette deux dés : X est la somme des nombres affichés par les dés.
2. On jette un dé jusqu'à obtenir 6 : X est le nombre de lancers réalisés.
3. On a appris un classifieur f, on teste f sur un nouvel exemple (x, y). X prend la valeur 1 si $f(x) \neq y$ et 0 sinon.
4. Idem. On tire un nouvel échantillon T. X est le risque empirique $R^T_{emp}(f)$ de f calculé sur T.

Définition 1. La fonction de répartition d'une variable aléatoire X est la fonction $F : \mathbb{R} \to [0, 1]$ définie par $F(x) = P(X < x)$.

Propriétés :
- F est croissante
- $\lim_{x\to+\infty} F(x) = 1$
- $\lim_{x\to-\infty} F(x) = 0$
- $P(X \in [a, b[) = F(b) - F(a)$

On dit qu'une variable aléatoire X est *continue* s'il existe une fonction de densité f telle que

$$P(X \in I) = \int_I f(x)dx.$$

Deux variables X et Y sont *indépendantes* si pour tous intervalles I et J,

$$P(X \times Y \in I \times J) = P(X \in I)P(Y \in J).$$

L'*espérance* $E(X)$ d'une variable aléatoire est la moyenne des valeurs prises par X. Elle est définie par :

$$E(X) = \sum_{\omega\in\Omega} P(\omega)X(\omega) = \sum_{a\in X(\Omega)} aP(X = a)$$

pour une variable finie ou dénombrable et par

$$E(X) = \int_{\mathbb{R}} xf(x)dx$$

pour une variable continue X de fonction de densité f.

Propriétés :
- Si X est une variable aléatoire constante prenant la valeur a, $E(X) = a$.

– E est une forme linéaire sur l'espace vectoriel des variables aléatoires : $E(aX+bY) = aE(X) + bE(Y)$ pour toutes v.a. X et Y et tous réels a et b.
– Si X et Y sont indépendantes, $E(XY) = E(X)E(Y)$.
– Inégalité de Markov : pour tout réel t,

$$P(X \geq t) \leq \frac{E(X)}{t}.$$

La *variance* d'une variable aléatoire mesure sa dispersion autour de sa valeur moyenne. Elle est définie par

$$V(X) = E((X - E(X)^2).$$

L'*écart-type* d'une v.a. est la racine carrée de sa variance :

$$\sigma(X) = \sqrt{V(X)}.$$

On peut interpréter l'écart type comme une distance dans un espace euclidien.

Propriétés :

– $V(X) = E(X^2) - E(X)^2$.
– $V(aX) = a^2V(X)$.
– Si X et Y sont des v.a. indépendantes, $V(X + Y) = V(X) + V(Y)$.
– Inégalité de Bienaymé-Tchebitcheff :

$$P(|X - E(X)| > t\sigma) \leq \frac{1}{t^2}$$

où t est une réel quelconque non nul.
Démonstration : $P(|X-E(X)| > t\sigma) = P((X-E(X))^2 > t^2V(X)) \leq V(X)/(t^2V(X)) = 1/t^2$ d'après l'inégalité de Markov. Cette inégalité est remarquable : elle mesure la concentration de toute variable aléatoire autour de son espérance. La contrepartie de son universalité est que cette inégalité est très faible.

On est souvent amené à considérer n variables aléatoires $X_1, \ldots, X_n$ indépendantes et de mêmes lois, ayant μ pour espérance et σ pour écart type. On définit alors la somme et la moyenne de ces variables par

$$S_n = X_1 + \ldots + X_n \text{ et } \overline{X} = \frac{X_1 + \ldots + X_n}{n}.$$

On déduit immédiatement des formules précédentes que

$$E(S_n) = n\mu, V(S_n) = n\sigma^2, E(\overline{X}) = \mu \text{ et } V(\overline{X}) = \frac{\sigma^2}{n}.$$

On déduit de l'inégalité de Bienaymé-Tchebitcheff que pour tout $\epsilon > 0$,

$$P(|\overline{X} - \mu| > \epsilon) \leq \frac{\sigma^2}{n\epsilon^2} \to 0 \text{ quand } n \to \infty.$$

Par conséquent (*loi faible des grands nombres*),

$$\lim_{n\to\infty} \frac{X_1 + \ldots + X_n}{n} = \mu \text{ avec probabilité 1.}$$

3.3 Quelques lois utiles

3.3.1 Loi de Bernoulli

La *loi de Bernoulli de paramètre* p est la loi d'une variable X prenant ses valeurs dans $\{0,1\}$ et telle que $P(X=1)=p$. On a $E(X)=p$ et $V(X)=p(1-p)$.

Exemple : Etant donné un classifieur f, la v.a. qui consiste à tirer un nouvel exemple (x,y) et retourner 1 si $y \neq f(x)$ et 0 sinon suit une loi de Bernoulli de paramètre $R(f)$.

3.3.2 Loi binomiale

La *loi binomiale de paramètres* n *et* p est la loi suivie par la somme $S = X_1 + \ldots + X_n$ de n variables de Bernoulli $X_1, \ldots, X_n$, indépendantes et de même paramètre p. S prend des valeurs entières, comprises entre 0 et n. Pour $0 \leq k \leq n$, on a $P(S=k) = C_n^k p^k (1-p)^{n-k}$. En appliquant les propriétés de l'espérance et de la variance, on obtient facilement $E(S) = np$ et $V(S) = np(1-p)$. Notons également qu'on a

$$E\left(\frac{X_1+\ldots+X_n}{n}\right) = p \text{ et } V\left(\frac{X_1+\ldots+X_n}{n}\right) = \frac{p(1-p)}{n}.$$

Exemple : Etant donné un classifieur f, la v.a. qui consiste à tirer n nouveaux exemples et retourner le nombre d'erreurs commises par f suit une loi binomiale de paramètres n et $R(f)$.

3.3.3 Loi normale

On appelle loi normale de paramètre (μ, σ) (ou encore loi de Laplace-Gauss) la loi d'une variable aléatoire continue de densité

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right).$$

Elle est notée $N(\mu, \sigma)$.

Propriétés :

– Si X suit une loi normale $N(\mu,\sigma)$, on a $E(X)=\mu$ et $V(X)=\sigma^2$.

– Si X suit une loi normale $N(0,1)$, la variable $\sigma X + \mu$ suit une loi $N(\mu, |\sigma|)$.

Le théorème central-limite :

Soit (X_n) une suite de variables aléatoires indépendantes de même loi, d'espérance μ et d'écart-type σ, alors

$$\sqrt{n}\left(\frac{\frac{X_1+\ldots+X_n}{n} - \mu}{\sigma}\right) \rightarrow^{\mathcal{L}} N(0,1)$$

où $X \rightarrow^{\mathcal{L}} Y$ signifie que la fonction de répartition de X converge uniformément vers celle de Y. On parle de *convergence uniforme en loi.*

On en déduit en particulier que la loi normale $B(n,p)$ de paramètres n et p converge vers la loi normale $N(np, \sqrt{np(1-p)})$. De même, si chaque X_i suit une loi de Bernoulli

de paramètre p, leur moyenne $(X_1 + \ldots + X_n)/n$ suit asymptotiquement une loi normale de paramètres p et $\sqrt{p(1-p)/n}$. En pratique, on identifie les deux lois dès que $n \geq 30$ et que $np(1-p) \geq 5$.

3.4 Estimation d'un paramètre

Un *estimateur* d'un paramètre θ est une variable aléatoire T dont les réalisations sont censées être des valeurs proches de θ.

Par exemple, le risque $R(f)$ d'un classifieur f est un paramètre. Un estimateur R_n de ce paramètre consiste à tirer n nouveaux exemples $(x_1, y_1), \ldots, (x_n, y_n)$ et à calculer la fréquence du nombre d'erreurs commises par f sur ces n exemples :

$$R_n = \frac{X_1 + \ldots + X_n}{n} \text{ où } X_i = 1 \text{ si } y_i = f(x_i) \text{ et 0 sinon.}$$

Fréquemment, un estimateur T dépend de n variables aléatoires $X_1, \ldots, X_n$ indépendantes et de même loi.

Le *biais* d'un estimateur est égal à $Biais(T) = E(T) - \theta$. On dit qu'un estimateur T du paramètre θ est *sans biais* si $E(T) = \theta$. Un estimateur $T = g(X_1, \ldots, X_n)$ est dit *convergent* si $\lim_{n\to\infty} g(X_1, \ldots, X_n) = \theta$.

Revenons à l'estimation de l'erreur d'un classifieur :

- R_n est une estimateur sans biais, pour toute valeur de n ; en particulier, R_1 est un estimateur sans biais.
- L'estimateur R_n, considéré comme dépendant de n, est convergent, d'après la loi des grands nombres.

D'une manière générale, si (X_n) est une suite de variables aléatoires de même loi et d'espérance μ, la variable aléatoire $S_n = \frac{X_1+\ldots+\ldots X_n}{n}$ est un estimateur sans biais (d'après la linéarité de l'espérance) et convergent (d'après la loi des grands nombres) de μ.

On peut également montrer que si (X_n) est une suite de variables aléatoires de même loi, d'espérance μ et de variance σ^2, la variable

$$V_n = \frac{\sum_{i=1}^{n}(X_i - S_n)^2}{n-1}$$

est un estimateur non biaisé et convergent de σ^2.

L'écart quadratique moyen $EQM(T)$ d'un estimateur T du paramètre θ mesure la dispersion de ses réalisations autour de la valeur θ. Il est définit par

$$EQM(T) = E(T - \theta)^2.$$

On montre que :

$$EQM(T) = V(T) + Biais(T)^2.$$

Démonstration :

$$\begin{aligned} EQM(T) &= E(T-\theta)^2 \\ &= E(T-E(T)+E(T)-\theta)^2 \\ &= E((T-E(T))^2 + E((E(T)-\theta)^2) + 2E(T-E(T))E(E(T)-\theta) \\ &= E((T-E(T))^2) + (E(T)-\theta)^2) + 2(E(T)-E(T))E(E(T)-\theta) \\ &= V(T) + Biais(T)^2. \end{aligned}$$

L'écart quadratique moyen d'un estimateur se décompose donc en la somme de deux termes : sa variance et le carré de son biais. Cela explique que l'on recherche souvent des estimateurs sans biais de variance minimale. Mais cette formule explique aussi pourquoi un estimateur biaisé mais peu dispersé peut être meilleur qu'un estimateur non biaisé.

Exemple : l'écart quadratique moyen de R_n est $R(f)(1-R(f)/n$. Tous les estimateurs R_n sont non biaisés mais on retrouve l'idée intuitive qu'une estimation de l'erreur sera d'autant meilleure qu'elle est calculée sur un grand nombre d'exemple.

3.5 Estimateur du maximum de vraisemblance

Soient $X_1, \ldots, X_n$ des variables aléatoires indépendantes de même loi dépendant de paramètres θ et soit $x_1, \ldots, x_n$ une réalisation de ces variables. La vraisemblance (likelihood) de ces observations est définie par

$$L(x_1, \ldots, x_n; \theta) = \prod_{i=1}^{n} P_\theta(X_i = x_i)$$

pour des variables discrètes et par

$$L(x_1, \ldots, x_n; \theta) = \prod_{i=1}^{n} f_\theta(x_i)$$

pour des variables continues de densité f_θ.

Exemple : on jette 3 fois une pièce dont la probabilité de tomber sur pile (resp. face) est p (resp. $1-p$). On observe : Pile, Pile, Face. La vraisemblance de cette observation est $p^2(1-p)$.

On définit également la *log-vraisemblance* par le logarithme de la vraisemblance :

$$logL(x_1, \ldots, x_n; \theta) = \sum_{i=1}^{n} \log P_\theta(X_i = x_i)$$

pour des variables discrètes et par

$$logL(x_1, \ldots, x_n; \theta) = \sum_{i=1}^{n} \log f_\theta(x_i)$$

pour des variables continues de densité f_θ.

On appelle estimateur du maximum de vraisemblance de θ la valeur de θ qui maximise la vraisemblance des observations. La fonction logarithme étant croissante, il est équivalent de maximiser la vraisemblance et la log-vraisemblance.

Exemple : on reprend l'exemple précédent. La log-vraisemblance de l'observation est $f(p) = 2\log p + \log 1 - p$. La dérivée de $f(p)$ est égale à

$$f'(p) = \frac{2}{p} - \frac{1}{1-p} = \frac{2-3p}{p(1-p)}.$$

On voit donc que cette fonction est croissante jusqu'en $p = 2/3$, puis décroissante. La vraisemblance est donc maximale pour $p = 2/3$.

D'une manière générale, si l'on observe m succès lors de l'observation d'une loi binomiale de paramètres n et p, l'estimateur du maximum de vraisemblance pour p est égal à m/n.

De même, si les variables indépendantes $X_1, \ldots, X_n$ suivent une loi normale $N(\mu, 1)$, la vraisemblance de l'observation $x_1, \ldots, x_n$ est

$$L(x_1, \ldots, x_n; \mu) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}} exp\left(-\frac{1}{2}(x_i - \mu)^2\right)$$

et

$$logL(x_1, \ldots, x_n; \mu) = -\frac{n}{2}\log 2\pi - \frac{1}{2}\sum_{i=1}^{n}(x_i - \mu)^2.$$

On montre facilement que la valeur de μ qui maximise la vraisemblance est

$$\mu = \sum_{i=1}^{n} x_i.$$

D'une ménière générale, on peut montrer que l'estimateur du maximum de vraisemblance est un estimateur convergent, asymptotiquement sans biais et asymptotiquement gaussien.

3.6 Intervalle de confiance d'un estimateur

Un intervalle de confiance au niveau α pour un paramètre θ est un intervalle aléatoire I, c'est-à-dire dont les bornes sont des variables aléatoires, tel que $P(\theta \in I) \geq \alpha$.

Considérons par exemple une variable X suivant une loi normale $N(\mu, \sigma)$. On peut démontrer que $P(\mu - 1,96\sigma < X < \mu + 1,96\sigma) \simeq 0,95$[5]. On en déduit que

$$P(X - 1,96\sigma < \mu < X + 1,96\sigma) \simeq 0,95.$$

[5]D'autres valeurs remarquables sont $P(\mu - 1,64\sigma < X < \mu + 1,64\sigma) \simeq 0,90$ et $P(\mu - 3,09\sigma < X < \mu + 3,09\sigma) \simeq 0,998$. Des tables fournissent des couples de valeurs (α, u_α) telles que $P(X - \mu > u_\alpha\sigma) < \alpha$. Etant donné la symétrie de la loi normale, on en déduit que $P(|X - \mu| > u_\alpha\sigma) < 2\alpha$ où encore que $P(|X - \mu| \leq u_\alpha\sigma) \geq 1 - 2\alpha$.

Soit x une réalisation de X : l'intervalle $[x-1,96\sigma, x+1,96\sigma]$ est un intervalle de confiance de μ au niveau de confiance de 95%.

Lorsque σ n'est pas connu, il faut l'estimer, par exemple à l'aide de l'estimateur V_n.

Considérons par exemple une suite (X_n) de variables aléatoires indépendantes suivant une loi normale de paramètres μ et σ. Soit $x_1, \ldots, x_n$ une réalisation de $X_1, \ldots, X_n$. Posons

$$\hat{\mu} = \frac{x_1 + \ldots + x_n}{n} \text{ et } \hat{\sigma} = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \hat{\mu})^2}{n-1}}.$$

L'intervalle $[\hat{\mu} - 1,96\hat{\sigma}, \hat{\mu} + 1,96\hat{\sigma}]$ est un intervalle de confiance approché de μ au niveau de confiance de 95%.

Reprenons maintenant l'exemple du classifieur f dont nous cherchons à estimer l'erreur $R(f)$. Supposons que sur un échantillon test composé de 1000 exemples, f commette 30 erreurs : 30 est la réalisation d'une loi binomiale de paramètres $R(f)$ et 1000. On en déduit que $R(f)$ peut être estimé par 30/1000=0,03. Avec quelle précision et quelle confiance ? La loi binomiale $B(n,p)$ peut être identifiée à une loi normale $N(np, \sqrt{np(1-p)})$ puisque $1000 \geq 30$ et $1000 \cdot 0,03 \cdot (1-0,03) \simeq 29,1 \geq 5$. L'écart-type de ces lois peut être estimé en utilisant l'estimation $\hat{p} = 0,03$ de p. On trouve : $\sqrt{1000 \cdot 0,03 \cdot 0,97} = 5,39$.

On en déduit donc que $[30 - 1,96 \cdot 5,39, 0,03 + 1,96 \cdot 5,39] \simeq [19,44; 40,56]$ est un intervalle de confiance approché de $nR(f)$ au niveau de confiance de 95%, où encore, que $[0,019; 0,041]$ est un intervalle de confiance approché de $R(f)$ au niveau de confiance de 95%.

D'une manière générale, si le classifieur f a classé incorrectement y exemples parmi un échantillon test comprenant n nouveaux exemples, et si $P(|X - \mu| \leq u_\alpha \sigma) \geq 1 - 2\alpha$ pour une variable X suivant une loi normale $N(\mu, \sigma)$,

$$\left[\frac{y}{n} - u_\alpha \sqrt{\frac{y(n-y)}{n^3}}, \frac{y}{n} + u_\alpha \sqrt{\frac{y(n-y)}{n^3}}\right]$$

est un intervalle de confiance approché de $R(f)$ au niveau de confiance $1 - 2\alpha$.

3.7 Exercices

1. Calculez l'espérance des 4 exemples de variables aléatoires définies dans la section 3.2.
2. Les trois portes
3. Détection de fraudes.
4. Le nombre de chars.

4 Bibliographie

Références

[AK94] E. Amaldi and V. Kann. On the approximability of finding maximum feasible subsystems of linear systems. *Lecture Notes in Computer Science*, 775 :521–? ?, 1994.

[BFOS84] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. Classification and regression trees. Technical report, Wadsworth International, Monterey, CA, 1984.

[BR92] A. L. Blum and R. L. Rivest. Training a 3-node neural network is NP-complete. *Neural Networks*, 5(1) :117–127, 1992.

[Bre01] Leo Breiman. Random forests. *Machine Learning*, 45(1) :5–32, 2001.

[CM02] Antoine Cornuejols and Laurent Miclet. *Apprentissage artificiel*. Eyrolles, 2002.

[Cyb89] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2(4) :303–314, 1989.

[EMS97] Floriana Esposito, Donato Malerba, and Giovanni Semeraro. A comparative analysis of methods for pruning decision trees. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(5) :476–491, 1997.

[HSH95] Klaus-U. Höffgen, Hans-U. Simon, and Kevin S. Van Horn. Robust trainability of single neurons. *Journal of Computer and System Sciences*, 50(1) :114–125, February 1995.

[HTF01] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical Learning*. Springer, 2001.

[JP78] D. S. Johnson and F. P. Preparata. The densest hemisphere problem. *Theoretical Computer Science*, 6(1) :93–107, February 1978.

[Mit97] T. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[MS63] J. N. Morgan and J. A. Sonquist. Problems in the analysis of survey data, and a proposal. *Journal of the American Statistical Association*, pages 415–434, 1963.

[Qui93] J. R. Quinlan. *C4.5 : Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[Qui94] *Comparing connectionist and symbolic learning methods*, volume I : Constraints and Prospects, chapter 15, pages 445–456. MIT Press, 1994.

[Sap90] G. Saporta. *Probabilités, analyse des données et statistiques*. Editions Technip, 1990.

[WF00] Ian H. Witten and Eibe Franck. *Data Mining : Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, 2000.