SY09 - TP4
Discrimination

Pierre Bathellier et Vançon Virgile

Université de Technologie de Compiègne

Introduction

Dans ce dernier TP nous allons aborder diverses notions de classification. Dans un premier temps notre étude se portera sur l'analyse discriminante. Pour cela nous étudierons les trois exemples vus pendant le semestre : l'analyse discriminante quadratique, linéaire ainsi que le classifieur bayésien naïf. Dans une seconde partie nous nous concentrerons sur la régression logistique. Pour chacune de ces méthodes, des tests itératifs seront effectués sur différents jeux de données afin d'étudier le comportement de ces classifieurs en fonction de la répartition des données.

1 Programmation

1.1 Analyse discriminante

Les analyses discriminantes quadratiques et linéaires ainsi que le classifieur bayésien naïf supposent, conditionnellement à chaque classe ω_k, que le vecteur de caractéristiques X suit une loi normale multidimensionnelle d'espérance μ_k et de variance Σ_k. Ces différents classifieurs sont issues de la théorie bayésienne de la décision, et ont pour objectif de minimiser une fonction de risque. Des hypothèses émises sur les paramètres de ces lois permettront d'obtenir différentes expressions de la règle de Bayes, qui permettront d'obtenir des règles de décision en remplaçant les paramètres théoriques par leurs estimations.

L'objectif de cette première partie est d'implémenter trois fonctions d'apprentissage et une fonction de calcul des probabilités à posteriori pour un ensemble de données.

Analyse discriminante quadratique

On parle d'analyse discriminante quadratique lorsque la distribution de X dans chaque classe ω_k suit des lois normales multivariées de paramètres μ_k et Σ_k différents. Dans ce cas, les fonctions $g_k(x)$ qui servent à définir la règle de décision ont des formes quadratiques. Les frontières de décision auront alors la forme de cercles, ellipses, paraboles, etc... lorsque l'on travaille en deux dimensions comme c'est notre cas.

Pour rappel, les densités conditionnelles f_k ont pour expression :

$$f_k(x) = \frac{1}{(2\pi)^{\frac{p}{2}}(det\Sigma_k)^{\frac{1}{2}}} exp(-\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k))$$

La règle de Bayes s'écrit alors :

$$\delta^(x) = a_k*$$

$$\text{avec } k^* = \underset{k}{argmax}\mathbb{P}(\omega_k|x) = \underset{k}{argmax} g_k(x)$$

Sachant que :

$$g_k(x) = ln(f_k(x)) + ln(\pi_k)$$

Soit :

$$g_k(x) = -\frac{1}{2}(x-\mu_k^T \Sigma_k^{-1}(x-\mu_k) - \frac{1}{2}ln(det\Sigma_k) + ln(\pi_k) - \frac{p}{2}2\pi)$$

L'objectif de la fonction d'apprentissage *adq.app* est donc de déterminer, partir du tableau données $Xapp$ et du vecteur des étiquettes associées *zapp*, les estimateurs du maximum de vraisemblance des paramètres $\widehat{\pi_k}$, $\widehat{\mu_k}$ et $\widehat{\Sigma_k}$:

$$\widehat{\pi_k} = \frac{n_k}{n}$$

$$\widehat{\mu_k} = \frac{1}{n_k}\sum_{i=1}^{n} z_{ik} i$$

$$\widehat{\Sigma_k} = V_k = \frac{1}{n_k}\sum_{i=1}^{n} z_{ik}(x_i - \widehat{\mu_k})(x_i - \widehat{\mu_k})^T$$

Dans les équations ci-dessus, z_{ik} représente une variable binaire d'appartenance de x à la classe ω_k. On utilisera, dans ce TP, l'estimateur sans biais $V_k^* = \frac{n_k}{n_k - 1}$.

Les frontières de décision obtenues pour les niveaux 0.2, 0.4, 0.6 et 0.8 pour le classifieur quadratique sont présentées figure 1

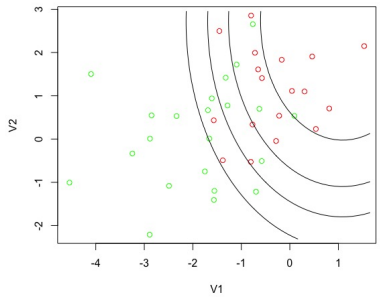

Fig. 1. Frontières de décisions 0.2, 0.4, 0.6 et 0.8 - Classifieur quadratique sur le jeu de donnée Synth1-40

Analyse discriminante linéaire

Dans le cas de l'analyse discriminante linéaire, la matrice de variance est commune à toutes les classes, c'est-à-dire que $\Sigma_k = \Sigma \forall k \in 1, ..., g$. Les fonctions discriminantes s'écrivent donc sous la forme d'équations linéaires de la forme :

$$g_k(x) = -\frac{1}{2}(x - \mu_k)^T \Sigma^{-1}(x - \mu_k) + ln(\pi_k)$$

On peut que le terme $(x - \mu_k)^T \Sigma^{-1}(x - \mu_k)$ est le carré de la *distance de Mahalanobis* de x à μ_k. Lorsque les probabilités a priori sont les mêmes, la règle de Bayes revient donc à affecter x à la classe dont le centre est le plus proche au sens de la *distance de Mahalanobis*.

La fonction d'apprentissage *adl.app* a pour but calculer les estimateurs du maximum de vraisemblance des paramètres $\widehat{\pi_k}$, $\widehat{\mu_k}$ (dont l'expression est la même que pour l'analyse discriminante quadratique) et $\widehat{\Sigma_k}$, dont la nouvelle expression est :

$$\widehat{\Sigma} = \frac{1}{n}\sum_{k=1}^{g} n_k V_k$$

On utilisera dans ce TP l'estimateur sans biais $V_W^* = \frac{1}{n-g}\sum_{k=1}^{g}(n_k - 1)V_k^*$

Les frontières de décision obtenues pour les niveaux 0.2, 0.4, 0.6 et 0.8 de l'analyse discriminante linéaire sont présentées figure 2

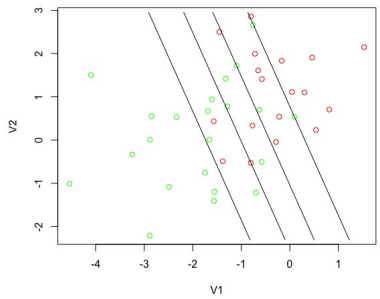

Fig. 2. Frontières de décisions 0.2, 0.4, 0.6 et 0.8 - Analyse discriminante linéaire sur le jeu de donnée Synth1-40

Classifieur bayésien naïf

En faisant l'hypothèse d'indépendance des variables X_j conditionnellement à Z, on peut supposer que les matrices Σ_k sont des matrices diagonales (dans le modèle gaussien). On obtient alors une variante de l'analyse discriminante quadratique, appelée classifieur bayésien naïf, dans laquelle les matrices Σ_k sont estimées par la matrice diagonale :

$$\widehat{\Sigma_k} = diag(s_{k1}^2, ..., s_{kp}^2)$$

Cette opération revient à annuler dans la matrice V_k les termes non-diagonaux. Les estimateurs $\widehat{\pi_k}$ et $\widehat{\mu_k}$ restent eux inchangés.

Les frontières de décision obtenues pour les niveaux 0.2, 0.4, 0.6 et 0.8 avec le classifieur bayésien naïf sont présentées figure 3

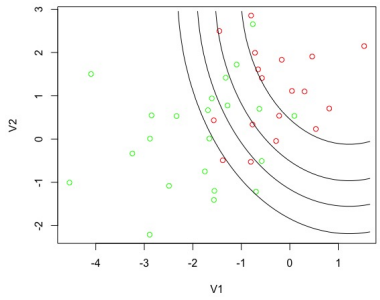

Fig. 3. Frontières de décisions 0.2, 0.4, 0.6 et 0.8 - Classifieur bayésien sur le jeu de donnée Synth1-40

Classification des données

La classification des données s'effectue grâce à la fonction *add.val*, qui calcule les probabilités à posteriori de chaque classe. En affectant x à la classe avec la plus grande probabilité a posteriori, règle de Bayes est respectée et la probabilité d'erreur$\epsilon(\delta)$ est minimisée. La fonction *add.val* s'appuie sur la fonction *mvdnorm*, qui permet de calculer la densité d'une loi normale multivariée pour un tableau de données. On rappelle que la probabilité a posteriori de la class ω_k peut être calculée comme selon la formule suivante :

$$\mathbb{P}(Z = \omega_k | X = x) = \frac{f_k(x)}{f(x)} = \frac{f_k(x)}{\pi_1 f_1(x) + \pi_2 f_2(x)}$$

1.2 Régression Logistique

Les analyses discriminantes quadratiques et linéaires, ainsi que le classifieur bayésien naïf, consistent à utiliser les fonctions de densités $f_k(x)$ dans lesquelles on remplace les paramètres par leurs estimateurs du maximum de vraisemblance (EMV). On utilise ensuite les valeurs des fonctions de densité pour calculer les probabilités a posteriori et ainsi affecter x à la classe ayant la probabilité la plus élevée. La régression logistique consiste à estimer directement les probabilités $\mathbb{P}(Z = \omega_k | X = x)$ d'appartenance aux classes.

L'objectif de cette partie est d'implémenter une fonction *log.app* chargée d'apprendre les paramètres du modèle et un fonction *log.val* chargée de classer un ensemble d'individus en fonction des résultats de *log.app*. Dans ce TP, n ous utiliserons le *modèle logit* qui consiste à exprimer le logarithme du ratio des probabilités à posteriori $\mathbb{P}(\omega_k|x)$ et $\mathbb{P}(\omega_g|x)$ comme un fonction de x pour tout $k = 1, ..., g-1$:

$$ln(\frac{\mathbb{P}(\omega_k|x)}{\mathbb{P}(\omega_g|x)}) = w_k^T x$$

On obtient dans le modèle général, l'expression des probabilités a posteriori des classes suivantes :

$$\mathbb{P}(\omega_g|x) = \frac{exp(w_k^T x)}{1 + \sum_{l=1}^{g-1} exp(w_l^T x)}$$

Dans le cas à deux classes ($g = 2$), on obtient :

$$p(x; w) = \mathbb{P}(\omega_1|x) = \frac{exp(w^T x)}{1 + exp(w^T x)}$$

Notre objectif est donc d'estimer les composantes du vecteur de poids w. Considérons une variable T_i codant l'information de classe z_i de la manière suivante :

$$t_i = \begin{cases} 1 \text{ si } Z_i = \omega_1 \\ 0 \text{ si } Z_i = \omega_2 \end{cases}$$

t_i peut alors être vu comme la réalisation d'un variable T_i suivant une loi binomiale de paramètre $p_i = p(x_i; w)$, c'est à dire : $T_i \sim \mathcal{B}(p_i)$. La fonction de log-vraisemblance associée à cette variable est :

$$ln(L(w; t_1, ..., t_n)) = \sum_{i=1}^{n} (t_i ln(p_i) + (1 - t_i) ln(1 - p_i))$$

Le gradient de cette fonction de log-vraisemblance s'écrit donc :

$$\frac{\partial lnL(w)}{\partial w} = X^T (t - p)$$

Avec $p = (p_1, ..., p_n)^T$ et $t = (t_1, ..., t_n)^T$. On cherche alors à résoudre le système à $p + 1$ équations, issu de l'équation de vraisemblance suivante :

$$\frac{\partial lnL(w)}{\partial w} = 0$$

Pour cela, il faut bien-sûr chercher le vecteur w maximisant la fonction $ln(L)$, car ce système ne peut pas être résolu directement. On utilise alors l'algorithme de *Newton-Raphson*, qui est un algorithme itératif consistant à faire, à la q^{me} itération, un développement limité de la fonction à maximiser (donc $L(w)$ dans notre cas) au voisinage de l'estimation courante $w^{(q)}$. Cette méthode consiste à sélectionner un vecteur de poids initial $w^{(0)}$, puis à calculer une nouvelle séquence de vecteurs $w^{(1)}$, $w^{(2)}$, etc... qui converge vers un maximum local de la log-vraisemblance. On arrêtera l'algorithme lorsque la différence entre 2 estimations successives β^q et β^{q+1} sera plus petite qu'un seuil ϵ, fixé à $1e-5$ dans ce TP. Pour effectuer ces calculs, nous avons besoin des coefficients de la matrice hessienne :

$$\frac{\partial^2 lnL(w)}{\partial w \partial w^T} = -\sum_{i=1}^{n} x_i x_i^T p_i (1 - p_i) = -X^T W X$$

où W est la matrice diagonale de terme général $W_{ii} = p_i(1 - p_i)$.
La règle de mise à jour des poids est alors :

$$w^{(q+1)} = w^{(q)} + (X^T W_{(q)} X)^{-1} X^T (t - p^{(q)})$$

Les frontières de décision obtenues pour les niveaux 0.2, 0.4, 0.6 et 0.8 en appliquant la régression logistique sont présentées figure 4.

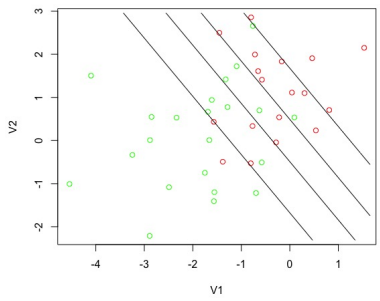

Fig. 4. Frontières de décisions 0.2, 0.4, 0.6 et 0.8 - Regression logistique sur le jeu de donnée Synth1-40

Régression logistique quadratique

La régression logistique quadratique est une généralisation simple du modèle de régression logistique, dans lequel on transforme nos données pour obtenir un espace plus complexe. Le modèle construit de cette manière sera plus flexible mais mais pourra s'avérer moins robuste car le nombre de paramètres à estimer est plus important. Dans la pratique, on effectue les produits des variables décrivant les individus d'apprentissage et on

apprend ensuite le modèle logistique sur $\frac{p(p+3)}{2}$ nouvelles variables. Par exemple, la matrice suivante décrivant les individus d'apprentissage :

$$X = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

deviendra :

$$X_2 = \begin{pmatrix} 1 & 2 & 2 & 1 & 4 \\ 3 & 4 & 12 & 9 & 16 \end{pmatrix}$$

Les frontières de décision obtenues pour les niveaux 0.2, 0.4, 0.6 et 0.8 en appliquant la régression logistique quadratique sont présentées figure 4.

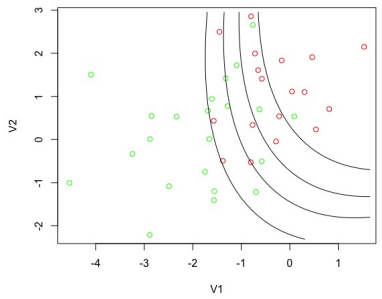

Fig. 5. Frontières de décisions 0.2, 0.4, 0.6 et 0.8 - Regression logistique sur le jeu de donnée Synth1-40

2 Application

Nous allons maintenant tester nos fonctions d'analyse discriminante et de régression logistiques sur plusieurs jeux de données.

Synth1-1000, **Synth2-1000** et **Synth3-1000** sont trois jeux de données simulées différents. Les données suivent dans chaque classe une loi normale multivariée.

Pima et **BreastCancer** sont des jeux de données réels. Pima correspond au diagnostic de diabète chez une population amérindienne. Les données BreastCancer correspondent à la prédiction du niveaux de gravité d'une tumeur à partir de certains descripteurs.

2.1 Test sur données simulées

Nous allons pour commencer tester nos fonctions sur les données simulées. Les données Synth1-1000, Synth2-1000 et Synth3-1000 ont été simulées selon des des loi multinomiales.

Le protocole est le même que le TP précedent :

Nous allons répéter 20 fois les classifications des différents jeux de données mis à notre disposition. A chaque séparation puis classification i, nous allons pouvoir déterminer une estimation du taux d'erreur E_i. La moyenne de ces taux d'erreur est un estimateur de ϵ. Pour chaque classifieur nous allons analyser la frontière de décision obtenues. Pour cela nous pourrons nous appuyer sur les fonctions prob.ad, prob.log et prob.log2. Les frontières présentées ont été déterminées avec un niveau à 0.5

On peut calculer des intervalles de confiance comme lors du TP précédent. On rappelle que l'on considère que le taux d'erreur ϵ suit la loi normale suivante :

$$\epsilon \sim \mathcal{N}(\mu, \sigma^2)$$

Ici μ, σ^2 sont inconnus mais peuvent être estimer respectivement par $\overline{X}$ qui est la moyenne empirique et par $S*^2$, qui est la variance empirique corrigé. La formule de l'intervalle de confiance est alors :

$$IC = \left[\overline{X} - t_{n-1;1-\frac{\alpha}{2}} \frac{S*}{\sqrt{n}}; \overline{X} + t_{n-1;1-\frac{\alpha}{2}} \frac{S*}{\sqrt{n}}\right]$$

Notons que pour les résultats lors des régressions logistiques classiques et quadratiques nous fixons la variable binaire *intr* à *true*.

En fixant cette variable à *true* on ajoute une ordonnée à l'origine à la matrices des exemples w. En pratique cela permet à la frontière de décision de ne pas passer par l'origine.

Les taux d'erreur que l'on obtient en appliquant cette méthode sont systématiquement meilleurs.

Données Synth1-1000

Résultats: La table 1 présente les taux d'erreur moyens et leurs intervalles de confiance pour N=20 tests de chaque classifieur sur le jeu de données Synth1-1000.

Table 1. Estimation des taux d'erreur de chaque classifieur sur le jeu de données Synth1-1000

	Adq	Adl	Nba	Logistique	Log quadratique	Arbre
ϵ	0.029	0.045	0.043	0.033	0.032	0.050
IC	[0.024;0.033]	[0.042;0.049]	[0.040;0.046]	[0.030;0.036]	[0.027;0.037]	[0.041;0.059]

Pour ce premier jeu de données à classifier nous fournissions figure 6, 7, 8, 9, 10 les frontières de décision obtenues pour chaque classifieur.

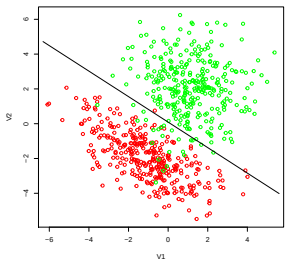

Fig. 6. Frontière de décision - classifieur adl - Synth1-1000

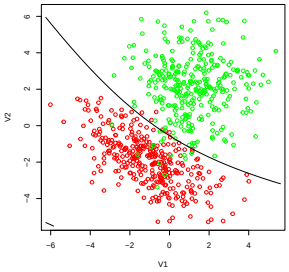

Fig. 7. Frontière de décision - classifieur adq - Synth1-1000

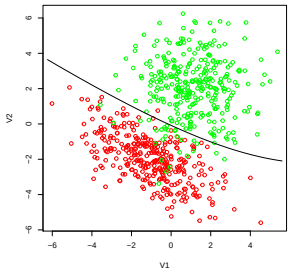

Fig. 8. Frontière de décision - classifieur nba - Synth1-1000

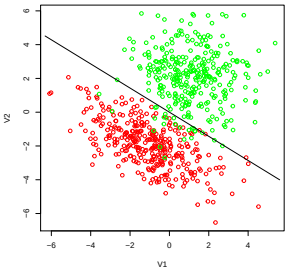

Fig. 9. Frontière de décision - Regression logistique classique - Synth1-1000

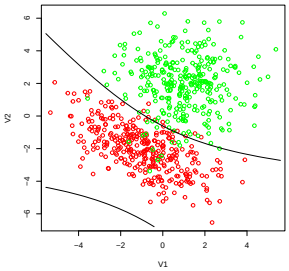

Fig. 10. Frontière de décision - regression logistique quadratique - Synth1-1000

Interprétation Les taux d'erreur obtenus sont relativement bas si on les compare aux résultats obtenues lors du TP précédent. Le classifieur le plus efficace sur ce jeu de donnée semble être le classifieur quadratiques et l'ADL semble être la méthode de classification la moins efficace. L'ADQ prend donc mieux en compte la différence de répartition entre les nuages de points des deux classes, l'ADL au contraire qui considère que les deux classes ont une matrice de variance commune obtient de mauvais résultats. L'hypothèse d'homoscédasticité sur laquelle repose l'ADL n'est donc pas vérifiée. Les estimateurs Σ_k des deux classes confirme ce raisonnement:

$$\Sigma_1 = \begin{pmatrix} 3 & -1.5 \\ -1.5 & 2 \end{pmatrix} \text{ et } \Sigma_2 = \begin{pmatrix} 2 & 0 \\ 0 & 3 \end{pmatrix}$$

. Le classifieur logistique classique semble lui aussi assez performant. Nous n'avons pour l'instant pas d'autres tests de ces classifieurs sur d'autres jeux de données pour déterminer la qualité de ces résultats. On rappelle que les résultats présentés pour les régressions logistiques classiques et quadratiques sont ceux en appliquant une ordonnée à l'origine à la matrice des exemples, les résultats sans l'ordonnée à l'origine pour la régression classique logistique et la régression logistique quadratiques sont respectivement de 0.043 et de 0.040. Ces résultats sont donc moins bons. L'ajout d'une ordonnée à l'origine permet à notre frontière de décision de ne pas passer par l'origine et donc d'avoir une forme épousant mieux les spécificités des nuages de points.

Données Synth2-1000

Résultats: La table 2 présente les taux d'erreur moyens et leurs intervalles de confiance pour N=20 tests de chaque classifieur sur le jeu de données Synth2-1000.

Table 2. Estimation des taux d'erreur de chaque classifieur sur le jeu de données Synth2-1000

	Adq	Adl	Nba	Logistique	Log quadratique	Arbre
ϵ	0.065	0.084	0.060	0.069	0.061	0.076
IC	[0.060;0.070]	[0.079;0.088]	[0.056;0.064]	[0.064;0.074]	[0.057;0.065]	[0.070;0.082]

Pour ce deuxième jeu de données nous avons choisi de ne pas présenter les 5 frontières de décision pour ne pas surcharger le rapport en graphiques. Nous présentons donc deux graphes qui sont assez représentatifs des différents résultats :

La frontière de décision de l'adq (figure) est représentative des frontières de décisions du nba et logistique quadratique, la frontière logistique classique (figure) est représentative de la frontière de décision de l'adl.

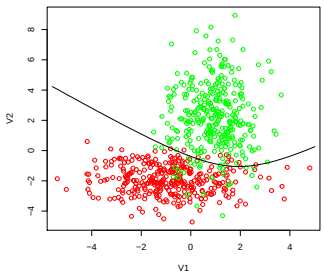

Fig. 11. Frontière de décision - classifieur adq - Synth2-1000

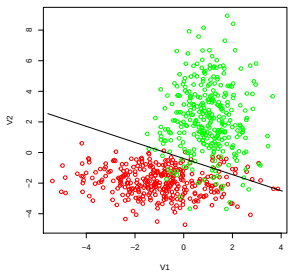

Fig. 12. Frontière de décision-logistique- Synth2-1000

Les nuages de points de points sont étalés selon une variable différente (La classe 1 en fonction de V2 et la classe 2 en fonction de V1).

Interprétation: La séparation des données entre les classes dans le jeu de données Synth2-1000 est moins nette. On constate une augmentation générale du taux d'erreur de chaque classifieur. Le classifieur adl semble encore une fois le moins performant. Les classifieurs quadratiques présentent des résultats légèrement meilleurs. Les mauvais résultats de l'ADL peuvent être expliqués de la même manière que pour le jeu de données Synth1-1000.

Données Synth3-1000

La table 3 présente les taux d'erreur moyens et leurs intervalles de confiance pour N=20 tests de chaque classifieur sur le jeu de données Synth3-1000.

Table 3. Estimation des taux d'erreur de chaque classifieur sur le jeu de données Synth3-1000

	Adq	Adl	Nba	Logistique	Log quadratique	Arbre
ϵ	0.043	0.041	0.048	0.039	0.042	0.058
IC	[0.040;0.046]	[0.037;0.045]	[0.041;0.054]	[0.035;0.043]	[0.038;0.046]	[0.051;0.065]

Pour ce dernier jeu de données, prenons les graphes des frontières de décision de la régression logistique classique et de la régression logistique quadratique (figure 13 et 14).

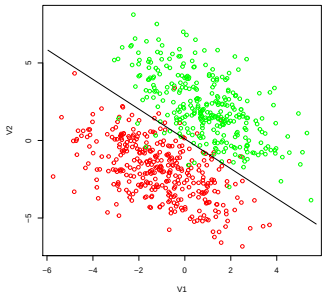

Fig. 13. Frontière de décision - logistique a - Synth2-1000

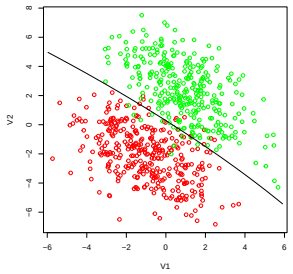

Fig. 14. Frontière de décision-logistique quadratique - Synth3-1000

Interprétation Les résultats que l'on obtient pour ce jeu de données sont très homogènes entre les différents classifieurs. Si l'on observe les frontières de décision on constate que les frontières des classifieurs quadratique, bayésien naïf et logistique qui ont normalement une forme ellipsoïdale sont très proches d'avoir une forme linéaire, la similitude des frontières de décision entre les classifieurs explique l'homogénéité globale des taux d'erreur. Dans ce cas l'ADL qui obtenait de mauvais résultats pour les jeux

2.2 Test sur données réelles

Données Pima

Résultats: La table 4 présente les taux d'erreur moyens et leurs intervalles de confiance pour N=20 tests de chaque classifieur sur le jeu de données pima.

Table 4. Estimation des taux d'erreur de chaque classifieur sur le jeu de données pima

	Adq	Adl	Nba	Logistique	Log quadratique	Arbre
ϵ	0.238	0.222	0.231	0.226	0.246	0.265
IC	[0.233;0.243]	[0.216;0.227]	[0.225;0.237]	[0.221;0.231]	[0.238;0.250]	[0.202; 0.332]

Un moyen de visualiser le jeu de donnée Pima est de représenter le jeu dans le premier plan factoriel de l'ACP (figure)

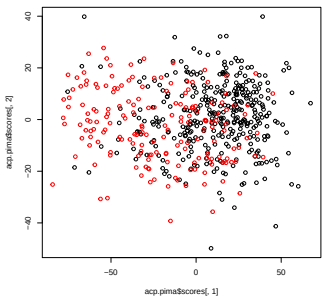

Fig. 15. Représentation du jeu de donnée pima dans le premier plan factoriel de l'acp (coloré par étiquette)

Interprétation: Les taux d'erreurs sont très élevés par rapport à ceux que l'on avait pu trouver sur les jeux de données simulées. La représentation de ce jeu de données dans le premier plan factoriel de l'ACP nous montre que les individus des deux classes sont difficilement distinguables. L'inertie inter classes est assez faible et qu'il est difficilement de définir une frontière séparant clairement les deux classes.

On rappelle que ces classifieurs sont appliqués en prenant l'hypothèses que le vecteur de caractéristique suit pour chaque classe une loi normale multivariée. Une explication de ces forts taux d'erreur serait que cette hypothèse faite ne soit pas vérifiée. En appliquant un test de Shapiro-Wilk (librairie mvShapiroTest) on obtient pour chaque classe une p-value $<$ 2.2e-16. Or le seuil d'acceptation est 0.05. On peut donc considérer que les deux classes ne suivent pas l'hyptothèse de répartition selon une loi normale multivariée. Comme dans les TP précedents on éprouve des difficultés dans la prédiction du diabète dans le jeu de données Pima, les variables explicatives de ce jeu de données sont plus difficilement exploitables que d'autres jeu de données, notamment BreastCancer.

Données BreastCancer

Résultats La table 5 présente les taux d'erreur moyens et leurs intervalles de confiance pour N=20 tests de chaque classifieur (sauf regression linéaire quadratique) sur le jeu de données BreastCancer.

Table 5. Estimation des taux d'erreur de chaque classifieur sur le jeu de données BreastCancer

	Adq	Adl	Nba	Logistique	Arbre
ϵ	0.048	0.045	0.038	0.041	0.056
IC	[0.045;0.050]	[0.043;0.047]	[0.036;0.040]	[0.038;0.043]	[0.026; 0.087]

Un moyen de visualiser le jeu de donnée BreastCancer est de représenter le jeu dans le premier plan factoriel de l'ACP (figure).

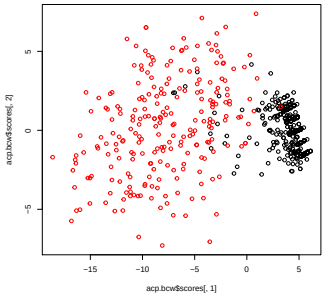

Fig. 16. Représentation du jeu de donnée BreastCancer dans le premier plan factoriel de l'acp (coloré par étiquette)

Interprétation: Les résultats sur ce jeu de donnée réel sont bien meilleurs sur ce jeu de donnée réel. Un indice sur cette performance est donnée par la représentation dans le premier plan de l'ACP. En effet les deux classes sont clairement distinguables. On suppose que les inerties inter classe sont bien plus élévées sur ce jeu de données. Les performances des classifieurs sont encore une fois assez homogènes, le classifieur Naïf baysien est toutefois le plus efficace. La performance de ce classifieur peut s'expliquer par le fait qu'il ne nécessite pas un grand nombre de données d'apprentissage.

3 Challenge : Données Spam

Dans cet exercice, on s'intéresse au jeu de données Spam, contenant 4601 ligne décrivant un ensemble de mails. On cherche ici a classifier ces mails et plus particulièrement à détecter des Spams, c'est à dire des messages indésirables. Chaque mail est caractérisé par 57 variables différentes et 1 étiquette indiquant son appartenance ou non à la classe "Spam". On peut raisonnablement supposer que chaque individu correspond à un message (classé ou non en tant que "Spam"), accompagné de différentes caractéristiques qui permettent de le classifier telles que l'adresse de l'émetteur, l'occurrence de certains mots, de majuscules, le nombre de liens, le nombre de fautes d'orthographe, etc... Comparativement au nombre d'individus, on est ici en présence d'un grand nombre de variables. De plus, certaines variables explicatives peuvent ne pas apporter d'information ou apporter une information induisant un mauvais classement. On a donc l'intuition que les différents modèles de prédiction n'auront pas de très bonnes performances.

Sachant que la détection de *Spams* est un problème de catégorisation de texte, nous avons recherché les différents algorithmes pouvant classifier au mieux ce type de données. Les principales méthodes utilisées dans la classification de texte sont le classifieur bayésien naïf, les méthodes reposant sur le *SVM* (Support Vector Machine), ou encore les méthodes basées sur des mesures de similarités cosinus (qui sont efficaces pour des grands ensembles de données). Les machines à vecteurs de support sont un ensemble de techniques d'apprentissage supervisé destinées à résoudre des problèmes de discrimination et de régression. Elles sont une généralisation des classifieurs linéaires.

Dans un premier temps, nous utilisons le classifieur bayésien naïf pour classifier nos données, avec 20 répétitions. Nous obtenons un taux d'erreur moyen de classification de $9,2\%$, ce qui est relativement important. On en conclue que ce classifieur ne permet pas de détecter les *Spams* avec une très bonne précision.

Nous appliquons alors les autres méthodes vues au cours du TP, et nous obtenons les résultats résumés dans le tableau suivant :

Méthode	Nba	Adq	Adl	Rdl	Rdq	Tree
Taux d'erreur moyen	9.2%	15.8%	16.2%	6.88%	7.75%	7.83%

Table 6. Estimation des taux d'erreur moyen de chaque classifieur sur le jeu de données Spam

Random Forest

Random Forest est un algorithme de machine learning qui est particulièrement efficace pour repérer des liens entre une variable à expliquer et des variables explicatives. Random Forest va classer les variables explicatives en fonction de leurs liens avec la variable à expliquer.

Une forêt aléatoire est un donc un ensemble d'arbres de décision binaire dans lequel a été introduit une part d'aléatoire. L'utilisation de forêts d'arbres aléatoires permet de calculer un ensemble d'arbres partiellement indépendants. Ce modèle réduit la variance des prévisions. L'inconvénient majeur de cette méthode est que l'on perd l'aspect visuel des arbres de décision uniques, qui rendait très clair le déroulement du processus de prédiction.

La construction de ces arbres utilise la technique du Bootstrap, c'est à dire que plutôt que d'utiliser toutes les données pour construire les arbres, on choisit aléatoirement un sous-ensemble des données pour chaque axe.

Nous avons choisi d'utiliser le modèle Random Forest :

- pour chaque arbre, on tire un échantillon à partir de l'échantillon initial
- à chaque noeud, on choisit aléatoirement K variables et on prend, parmi celles-ci, celle qui minimise le critère de l'algorithme CART
- on laisse grandir l'arbre jusqu'à ce qu'il n'y ait plus qu'un seul élément dans chaque noeud
-

La bibliothèque randomForest de R nous permet de réaliser facilement de tels arbres. La fonction *randomForest* nous renvoi notamment un objet qui quantifie l'impact de chaque variable explicative sur la prédiction par deux approches : la moyenne de la perte de précision associée à la perte de cette variable, et la moyenne de la chute de l'indice de Gini associé à cette variable. L'*indice de Gini* permet, rappelons le, de mesurer l'impureté

caractérisant l'homogénéité des sous-ensembles résultant d'une séparation. Ce critère est nul si tous les éléments sont bien classés, et est maximum si les éléments des différentes classes sont représentés en proportions égales dans l'ensemble. Il se calcule suivant la formule :

$$G(p) = \sum_{k=1}^{g} p_k(1 - p_k)$$

Nous avons décidé de ne garder que les variables dont l'impact sur l'*indice de Gini* était supérieur à 10. Cela a permit de conserver un ensemble de 28 variables explicatives (c'est-à-dire la moitié de ce que nous avions auparavent). En appliquant notre fonction de prédiction par arbre décisionnel implémentée précédemment à ce nouveau jeu de données, nous obtenons des performances très intéressantes, avec un taux d'erreur de 0.8% et un écart-type de 0.5%. La sélection de variables par *randomForest* offre donc une solution performante pour le jeu de données Spams.

Conclusion

Différentes méthodes de classificaion ont donc pu être observées et testées au cours de cet ultime TP, chacune présente ses avantages et ses défauts. Nous avons pu constater que leur application effective sur des ensemble de données peut être parfois complexe et peu performante. Ceci nous laisse ainsi, le rôle d'analyse, de décision et de calcul des risques, pour mener à bien des classifications sur de gros jeux de données, même avec des algorithmes et des méthodes statistiques puissantes.

List of Figures