

**SY09 Printemps 2015**  
**TP 4**  
**Analyse discriminante, régression logistique**

## 1 Analyse discriminante

### 1.1 Implémentation

On se propose d'étudier et comparer trois modèles d'analyse discriminante *dans le cas binaire* (jeux de données comptant  $g = 2$  classes) : l'analyse discriminante quadratique, l'analyse discriminante linéaire, et le classifieur bayésien naïf.

1. Rappeler les estimateurs des paramètres du modèle dans chacun de ces cas (on choisira les estimateurs sans biais).
2. Programmer les fonctions `adq.app`, `adl.app`, `nba.app` et `ad.val`.

Les trois premières font l'apprentissage des trois modèles d'analyse discriminante mentionnés ci-dessus : elles doivent donc prendre en argument d'entrée le tableau de données `Xapp` et le vecteur `zapp` des étiquettes associées, et retourner les paramètres du modèle (proportions, vecteurs de moyennes et matrices de covariance des deux classes) que l'on pourra stocker dans une structure quelconque.

La fonction `ad.val` calcule les probabilités a posteriori pour un ensemble de données, puis effectue le classement en fonction de ces probabilités : elle prend donc en compte les paramètres du modèle et l'ensemble de données `Xtst` à classer, et retourne une structure contenant les probabilités a posteriori `prob` estimées et le classement associé.

Cette fonction pourra s'appuyer sur la fonction `mvdnorm`, disponible sur le site de l'UV, qui permet de calculer la densité d'une loi normale multivariée pour un tableau de données.

On pourra utiliser la fonction `prob.ad`, disponible sur le site de l'UV, pour afficher les courbes de niveau des probabilités a posteriori  $\hat{\mathbb{P}}(\omega_1|\mathbf{x})$  estimées. La frontière de décision correspond à la courbe de niveau  $\mathbb{P}(\omega_1|\mathbf{x}) = 0.5$ .

### 1.2 Test sur données simulées

On souhaite comparer les performances de l'analyse discriminante sur les jeux de données simulées `Synth1-1000`, `Synth2-1000` et `Synth3-1000` disponibles sur le site de l'UV. Pour ce faire, on utilisera le même protocole expérimental que dans le TP précédent :

1. séparer le jeu de données en un ensemble d'apprentissage et un ensemble de test ;
2. apprendre les paramètres du modèle sur l'ensemble d'apprentissage,
3. effectuer le classement des données de test et calculer le taux d'erreur associé.

On répétera cette procédure  $N = 20$  fois pour chaque jeu de données.

Pour chaque jeu de données, calculer le taux d'erreur (de test) moyen sur les  $N = 20$  séparations effectuées. On pourra s'appuyer sur les frontières de décision obtenues pour analyser les résultats. Sachant que les données suivent dans chaque classe une loi normale multivariée, comment interprétez-vous ces résultats ?

## 2 Régression logistique

On souhaite appliquer le modèle logistique *binaire* sur les trois jeux de données de l'exercice 1. On programmera tout d'abord deux fonctions, l'une permettant de faire l'apprentissage du modèle (on utilisera l'algorithme de Newton-Raphson présenté en cours), l'autre permettant d'appliquer le modèle obtenu sur un ensemble de données.

### 2.1 Implémentation

**Apprentissage.** La fonction `log.app`, permettant d'apprendre les paramètres du modèles, prendra comme arguments d'entrée le tableau de données `Xapp`, le vecteur `zapp` des étiquettes associées, ainsi qu'une variable binaire `intr` indiquant s'il faut ou non ajouter une ordonnée à l'origine (*intercept*) à la matrice d'exemples et un scalaire `epsi` correspondant au seuil  $\varepsilon$  en-deçà duquel on considère que l'algorithme d'apprentissage a convergé.

Elle devra retourner la matrice `beta` correspondant à l'estimateur du maximum de vraisemblance  $\hat{\beta}$  des paramètres, de dimensions  $p \times 1$  (ou  $(p+1) \times 1$  si une ordonnée à l'origine a été ajoutée), le nombre `niter` d'itérations effectuées par l'algorithme de Newton-Raphson, et la valeur `logL` de la vraisemblance à l'optimum.

On pourra utiliser comme matrice de paramètres initiale  $\beta^{(0)} = (0, \dots, 0)^t$ . La convergence sera testée en comparant la norme de la différence entre deux estimations successives  $\beta^{(q)}$  et  $\beta^{(q+1)}$  au seuil  $\varepsilon$  (on pourra choisir  $\varepsilon = 1e-5$ ).

**Classement.** La fonction `log.val`, permettant d'évaluer un ensemble de test, prendra comme arguments d'entrée le tableau de données `Xtst` à classer et la matrice `beta` des paramètres déterminés par la fonction `log.app`. Cette fonction fera un test sur les dimensions de la matrice `beta`, pour déterminer si une ordonnée à l'origine doit être ou non ajoutée à la matrice d'exemples `Xtst` à évaluer.

Elle devra retourner une structure contenant la matrice `prob` des probabilités a posteriori estimées et le vecteur des classements associés.

Ces fonctions pourront s'appuyer sur une fonction calculant les probabilités a posteriori à partir d'une matrice de paramètres `beta` et d'un tableau de données `X`.

**Régression logistique quadratique.** Il est possible de généraliser le modèle de régression logistique de manière très simple. La stratégie consiste à transformer les données dans un espace plus complexe, dans lequel les classes peuvent être séparées par un hyperplan. La régression logistique est alors effectuée dans cet espace.

Par exemple, dans le cas où les individus sont décrits par les variables  $X^1$ ,  $X^2$  et  $X^3$ , la régression logistique quadratique consiste à effectuer la régression logistique classique dans l'espace correspondant aux variables  $X^1$ ,  $X^2$ ,  $X^3$ ,  $X^1X^2$ ,  $X^1X^3$ ,  $X^2X^3$ ,  $(X^1)^2$ ,  $(X^2)^2$  et  $(X^3)^2$ , que l'on notera  $\mathcal{X}^2$ , plutôt que dans l'espace  $\mathcal{X} = \{X^1, X^2, X^3\}$ .

Le modèle ainsi défini est donc plus flexible; mais le nombre de paramètres à estimer étant plus important ( $p(p+3)/2$  au lieu de  $p$ ), il peut également s'avérer moins robuste que le modèle classique déterminé dans l'espace des caractéristiques initiales.

### 2.2 Test sur données simulées

On souhaite comparer les performances des deux modèles de régression logistique sur les trois jeux de données simulées étudiés précédemment. Pour ce faire, on utilisera le même protocole expérimental que pour l'analyse discriminante : pour chaque jeu de données, calculer le taux d'erreur (de test) moyen sur les  $N = 20$  séparations effectuées. Qu'observe-t-on ? Comment interpréter ces résultats ?

On pourra utiliser les fonctions `prob.log` et `prob.log2`<sup>1</sup>, disponible sur le site de l'UV, pour visualiser les courbes de niveau ou les frontières de décision obtenues respectivement par régression logistique et par régression logistique quadratique.

---

1. On veillera à ce que l'ordre des termes quadratiques soit bien le même dans les fonctions développées et dans la fonction `prob.log2`.

## 3 Données réelles

### 3.1 Données « Pima »

On souhaite appliquer les trois modèles d'analyse discriminante et les deux modèles de régression logistique à la prédiction du diabète chez les individus d'une population d'amérindiens. On pourra charger les données au moyen du code suivant :

```
Donn <- read.csv("Pima.csv", header=T)
X <- Donn[,1:7]
z <- Donn[,8]
```

On utilisera ensuite le même protocole expérimental que pour les tests sur données simulées, en répétant l'expérience  $N = 100$  fois. Calculer les taux moyens d'erreur de test pour chacun des cinq modèles étudiés. Que constate-t-on ? Comment expliquez-vous ces résultats ?

Au cours d'une expérience<sup>2</sup>, on pourra utiliser le code suivant pour créer les données sur lesquelles apprendre le modèle de régression logistique quadratique :

```
Xapp2 <- Xapp
Xtst2 <- Xtst
for (p in 1:(dim(Xapp)[2]-1))
{
  for (q in (p+1):dim(Xapp)[2])
  {
    Xapp2 <- cbind(Xapp2, Xapp[,p]*Xapp[,q])
    Xtst2 <- cbind(Xtst2, Xtst[,p]*Xtst[,q])
  }
}
for (p in 1:(dim(Xapp)[2]-1))
{
  Xapp2 <- cbind(Xapp2, Xapp[,p]^2)
  Xtst2 <- cbind(Xtst2, Xtst[,p]^2)
}
```

Faire l'apprentissage des modèles que l'on souhaite comparer. Calculer les taux d'erreur de test correspondants. Que peut-on constater ? Comment expliquer ces résultats ?

### 3.2 Données « breast cancer Wisconsin »

On considère à présent un problème de prédiction du niveau de gravité d'une tumeur à partir de descripteurs physiologiques. On récupérera les données sur le site de l'UV et on les chargera en utilisant le code suivant :

```
Donn <- read.csv("bcw.csv", header=T)
X <- Donn[,1:9]
z <- Donn[,10]
```

On répétera l'expérience (séparation, apprentissage et évaluation des performances)  $N = 100$  fois. Calculer les taux moyens d'erreur de test pour les trois modèles d'analyse discriminante et la régression logistique classique (on n'utilisera pas la régression logistique quadratique ici). Que constate-t-on ? Interpréter et commenter.

---

2. On prendra bien garde de créer les variables quadratiques une fois faite la séparation du jeu de données en ensemble d'apprentissage et de test : dans le cas contraire (création avant séparation), les ensembles d'apprentissage et de test ne correspondraient pas aux mêmes individus.