SY09 - Rapport de projet

Projet 2

Han GAO - Lucie NOE

UTC - Printemps 2018

1 Discrimination

1.1 Programmation

Dans cette partie, on se vise à évaluer les performances des principales méthodes de discrimination étudiées dans le cours - analyse discriminante quadratique (ADQ) et linéaire (ADL), classifieur bayésien naïf (NBA), régression logistique, arbres de décision (AD) sur différents jeux de données binaires. Nous ferons quelques rappels sur les hypothèses et les estimateurs spécifiques à chaque méthode.
Pour ces classifieurs, on émet l'hypothèse suivante : la matrice X suit conditionnellement à chaque classe ω_k une loi normale multidimensionnelle d'espérance μ_k et de variance $\sum k$.

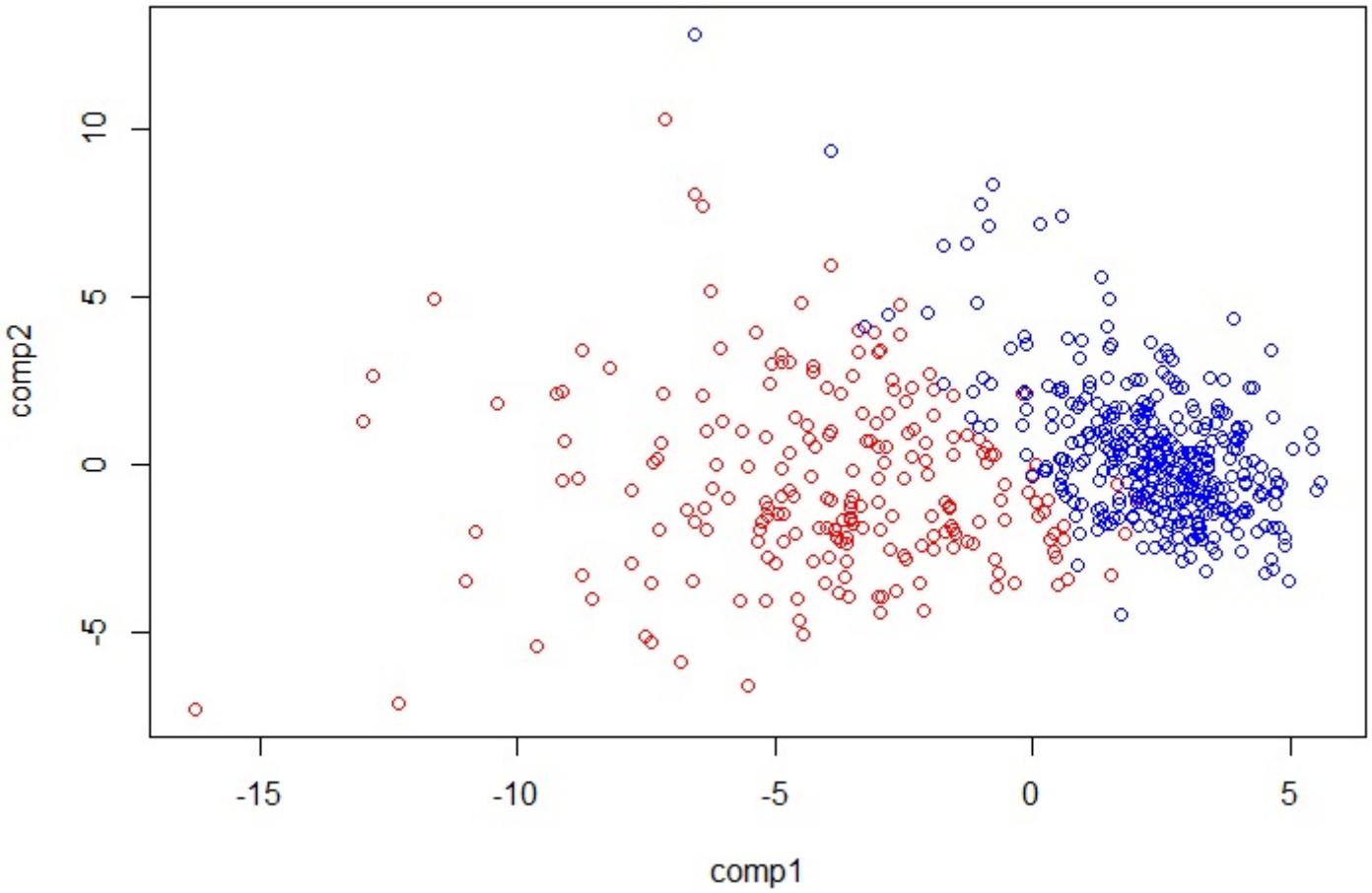
On complétera pour cela quatre fonctions : *adq.app*, *adl.app*, *nba.app*, *ad.val*. Le code de ces modèles d'analyse diacriminante, ainsi que la fonction éffectuant les prédictions en fonction des paramètres sont en annexe.

1.2 Application

Nous cherchons à comparer les performances de l'analyse discriminante, de la régression logistique et des arbres de décision sur les jeux de données binaires *breast-cancer*, *ionosphere*, *sonar*, *spambase*, *spambase2*. Pour ce faire, nous répéterons 100 fois le protocole suivant :
- Séparation aléatoire des données en un ensemble d'apprentissage et un ensemble de test
- Apprentissage du modèle sur l'ensemble d'apprentissage
- Classement des données de test et calculer le taux d'erreur associé. Il s'agit de faire la moyenne des 100 taux d'erreur obtenus comme suivantes.

Données breastcancer

Dans le premier temps, pour l'analyse discriminante linéair, nous n'avons effectué aucun pré-traitement de données avec N = 100 répétitions. En revanche, pour l'analyse discriminante quadratique et le classifieur bayésien, nous avons rencontré des difficultés. La matrice de variance contient des valeurs propres nulles ou très proches de 0. De ce fait, on a utilisé l'ACP, comme vu au premier projet, pour le prétraitement. On obtient le graphique suivant :



Nouc remarquons que l'estimation de l'erreur pour N = 100 répétitions de l'analyse discriminante est proche de la valeur ponctuelle trouvée précédemment sans pré-traitement de données.

Classifieur	Taux erreur moyen
ADQ	4.8
ADL	4.93
NBA	6.626
RL	4.716
ABD	7.45

Table 1.2.1 - Performance des classifications sur les données breastcancer pour 100 exécutions

Concernant le jeu de données *breastcancer*, nous obtenons ici de bons résultats. La méthode discriminante quadratique et la régression logistique ont la meilleur performance. Les paramètres à estimer pour ces deux méthodes sont les plus importantes, donc ces modèles ont plus de flexibilité que les autres. De manière plus générale, la performance des prédictions obtenus avec tous nos méthodes est très effective.

Données ionosphere

Classifieur	Taux erreur moyen
ADQ	13.53
ADL	15.41
NBA	18.29
RL	16.58
ABD	11.20

Table 1.2.2 - Performance des classifications sur les données ionosphere pour 100 exécutions.

Concernant le jeu de données *ionosphere*, on constate que les taux d'erreur sont plus élevés par rapport aux données précédentes : on en déduit qu'aucun classifieur ne permet de construire une frontière de décision suffisante pour distinguer les deux classes. On peut supposer que ce phénomène vient alors des données en elles mêmes.

Données spambase

Classifieur	Taux erreur moyen
ADQ	17.06
ADL	11.57
NBA	18.20
RL	7.29
ABD	10.23

Table 1.2.3 - Performance des classifications sur les données spambase pour 100 exécutions

Données spambase2

Classifieur	Taux erreur moyen
ADQ	11.68
ADL	7.267
NBA	11.085
RL	6.2
ABD	11.44

Table 1.2.4 - Performance des classifications sur les données spambase2 pour 100 exécutions

Concernant les jeu de données *spambase* et *spambase2*, la régression logistique présente les taux d'erreurs les plus petites. L'analyse discriminante linéaire fonctionne plutôt bien que l'analyse discriminante quadratique. Nous pouvons expliquer cela par le fait que le modèle linéaire comporte moins de paramètres que le modèle donc il est plus général et plus robuste sur de nouvelles données. En revance, le classifieur bayésien possède le moins de paramètres et n'est donc pas assez flexible pour ces jeux de données.

2 Modèle

Question 1

On suppose que pour chaque variable X^j, la proportion (théorique) p_{kj} de valeurs égales à 1 dépend de la classe :

$$p_{kj} = Pr(X^j = 1 \mid Z = \omega_k)$$

D'après les changements opérés pour passer de **spambase** à **spambase2**, les valeurs des x_{ij} ne peuvent que prendre les valeurs 1 ou 0.

On trouve donc que :

$$Pr(X^j = 0 \mid Z = \omega_k) = 1 - p_{kj}$$

On obtient donc une distribution conditionnelle suivant une loi de Bernoulli de paramètre p_{kj} :

$$\boxed{Pr(X^j = x_j \mid Z = \omega_k) = (p_{kj})^{x_j} \ (1-p_{kj})^{1-x_j}}$$

Question 2

$$Pr(X = x \mid Z = \omega_k) = Pr(X^1 = x_1 \cap X^2 = x_2 \cap ... \cap X^p = x_p \mid Z = \omega_k)$$

Par hypothèse, on a l'indépendance des variables $X^1, X^2, ..., X^p$ conditionnellement à la classe Z = ω_k.
On obtient donc :

$$Pr(X = x \mid Z = \omega_k) = \prod_{j=1}^{p} Pr(X^j = x_j \mid Z = \omega_k)$$

$$\boxed{\Leftrightarrow Pr(X = x \mid Z = \omega_k) = \prod_{j=1}^{p} (p_{kj})^{x_j} \ (1-p_{kj})^{1-x_j}}$$

Question 3

On considère que le i^e exemple d'apprentissage consiste en un couple $(x_i\,;\,z_i)$.
On cherche à écrire la probabilité jointe :

$$Pr(X = x_i,\ Z = z_i)$$

On peut noter :

$$Pr(X = x_i,\ Z = z_i) = Pr(X = x_i \mid Z = z_i)\ Pr(Z = z_i)$$

D'après l'hypothèse faite dans la question précédente, on a l'indépendance des variables $X^1, ..., X^p$ conditionnellement à la classe Z.

$$\boxed{\Leftrightarrow Pr(X = x_i,\ Z = z_i) = \prod_{j=1}^{p} Pr(X^j = x_{ij} \mid Z = z_i)\ Pr(Z = z_i)}$$

De plus, on sait que $z_i = (z_{i1}, ..., z_{ig})^T$, et que $z_{ik} = \mathbb{1}_{\omega_k}(x_i)$.
On peut donc écrire : $Pr(Z = z_i) = \prod_{k=1}^{g} Pr(Z^k = 1)^{z_{ik}} = \prod_{k=1}^{g} Pr(Z^k = \omega_k)^{z_{ik}} = \prod_{k=1}^{g} Pr(Z = \omega_k)^{z_{ik}}$ car ech. *iid.*

En suivant le même raisonnement, on obtient :

$$Pr(X^j = x_{ij} | Z = z_i) = \prod_{k=1}^{g} Pr(X^j = x_{ij} | Z = \omega_k)^{z_{ik}}$$

D'après la premère question :

$$Pr(X^j = x_{ij} | Z = z_i) = \prod_{k=1}^{g} ((p_{kj})^{x_{ij}} (1 - p_{kj})^{1-x_{ij}})^{z_{ik}}$$

A l'aide de ces équations, on peut trouver la forme de la probabilité jointe :

$$Pr(X = x_i, Z = z_i) = \prod_{j=1}^{p} (\prod_{k=1}^{g} ((p_{kj})^{x_{ij}} (1 - p_{kj})^{1-x_{ij}})^{z_{ik}}) \prod_{k=1}^{g} Pr(Z = \omega_k)^{z_{ik}}$$

$$\boxed{\Leftrightarrow Pr(X = x_i, Z = z_i) = \prod_{k=1}^{g} [Pr(Z = \omega_k)^{z_{ik}} \prod_{j=1}^{p} ((p_{kj})^{x_{ij}} (1 - p_{kj})^{1-x_{ij}})^{z_{ik}}]}$$

Question 4

On note $L((x_1, ..., x_p), (z_1, ..., z_p) | p_{kj}, \pi_k)$ la vraisemblance, avec $\pi_k = Pr(Z = \omega_k)$.

$$L((x_1, ..., x_p); (z_1, ..., z_p) | p_{kj}, \pi_k) = \prod_{i=1}^{n} Pr(X = x_i, Z = z_i)$$

D'après la dernière question :

$$\Leftrightarrow L((x_1,...,x_p);(z_1,...,z_p)|p_{kj},\pi_k) = \prod_{i=1}^{n}\prod_{k=1}^{g}[{\pi_k}^{z_{ik}}\prod_{j=1}^{p}((p_{kj})^{x_{ij}}(1-p_{kj})^{1-x_{ij}})^{z_{ik}}]$$

Question 5

A l'aide de la forme de la vraisemblance écrite ci-dessus, on cherche à trouver l'EMV des deux paramètres : π_k et p_{kj}.

Pour calculer le maximum de la vraisemblance, il faut trouver les valeurs pour lesquelles la dérivée de la vraisemblance s'annule. Afin de simplifier les calculs, on cherchera plutôt à maximiser le logarithme de la vraisemblance, ce qui revient au même puisque la fonction ln est croissante.

$$ln(L((x_1,...,x_p);(z_1,...,z_p)|p_{kj},\pi_k)) = \sum_{i=1}^{n}\sum_{k=1}^{g}[z_{ik}ln(\pi_k) + \sum_{j=1}^{p} z_{ik}(x_{ij}ln(p_{kj}) + (1-x_{ij})ln(1-p_{kj}))$$

- L'EMV de p_{kj} :

Soit $k \in [\![1,g]\!]$ et $j \in [\![1,p]\!]$.

$$\partial ln(L((x_1,...,x_p);(z_1,...,z_p)|p_{kj},\pi_k))\partial p_{kj} = 0$$

$$\Leftrightarrow \partial\partial p_{kj}(\sum_{i=1}^{n}\sum_{k=1}^{g}[z_{ik}ln(\pi_k) + \sum_{j=1}^{p} z_{ik}(x_{ij}ln(p_{kj}) + (1-x_{ij})ln(1-p_{kj}))]) = 0$$

Or on a fixé k et j :

$$\Leftrightarrow \partial\partial p_{kj}(\sum_{i=1}^{n}[z_{ik}ln(\pi_k) + z_{ik}(x_{ij}ln(p_{kj}) + (1-x_{ij})ln(1-p_{kj}))]) = 0$$

$$\Leftrightarrow \sum_{i=1}^{n} z_{ik}(\frac{x_{ij}}{p_{kj}} - \frac{1-x_{ij}}{1-p_{kj}}) = 0$$

$$\Leftrightarrow \sum_{i=1}^{n} z_{ik}(\frac{x_{ij}-p_{kj}}{p_{kj}-p_{kj}^2}) = 0$$

$p_{kj} - p_{kj}^2 = p_{kj}(1-p_{kj}) \neq 0$ car, étant une probabilité, $0 < p_{kj} < 1$.

$$\Leftrightarrow \sum_{i=1}^{n} z_{ik}(x_{ij} - p_{kj}) = 0$$

$$\Leftrightarrow \sum_{i=1}^{n} z_{ik}x_{ij} - \sum_{i=1}^{n} z_{ik}p_{kj} = 0$$

$$\Leftrightarrow \sum_{i=1}^{n} z_{ik}x_{ij} = p_{kj}\sum_{i=1}^{n} z_{ik}$$

En notant $n_k = \sum_{i=1}^{n} z_{ik}$, on obtient donc que la dérivée s'annule en $\boxed{p_{kj} = \frac{1}{n_k}\sum_{i=1}^{n} z_{ik}x_{ij}}$.

On calcule ensuite la dérivée seconde afin de s'assurer que la valeur de p_{kj} trouvée ci-dessus correspond bien à un maximum.

$$\frac{\partial^2 ln(L((x_1,...,x_p);(z_1,...,z_p)|p_{kj},\pi_k))}{\partial^2 p_{kj}} = \frac{\partial^2 \sum_{i=1}^{n} z_{ik}(\frac{x_{ij}-p_{kj}}{p_{kj}-p_{kj}^2})}{\partial^2 p_{kj}}$$

$$\Leftrightarrow \frac{\partial^2 ln(L((x_1,...,x_p);(z_1,...,z_p)|p_{kj},\pi_k))}{\partial^2 p_{kj}} = \sum_{i=1}^{n} z_{ik}\frac{\partial^2}{\partial^2 p_{kj}}(\frac{x_{ij}-p_{kj}}{p_{kj}-p_{kj}^2})$$

$$\Leftrightarrow \frac{\partial^2 ln(L((x_1,...,x_p);(z_1,...,z_p)|p_{kj},\pi_k))}{\partial^2 p_{kj}} = \sum_{i=1}^{n} z_{ik}\frac{-x_{ij}+2x_{ij}p_{kj}-p_{kj}^2}{(p_{kj}-p_{kj}^2)^2}$$

Or, puisque x ne peut prendre comme valeurs que 1 ouu 0 :

$$-x_{ij}+2x_{ij}p_{kj}-p_{kj}^2 = \begin{cases} -1-2p_{kj}-p_{kj} & \text{si } x=1 \\ -p_{kj} & \text{sinon.} \end{cases}$$

c'est-à-dire :

$$-x_{ij}+2x_{ij}p_{kj}-p_{kj}^2 = \begin{cases} -(1-p_{kj})^2 & \text{si } x=1 \\ -p_{kj} & \text{sinon.} \end{cases}$$

Puisque $0 < p_{kj} < 1$, on a bien une dérivée partielle seconde strictement négative.
La valeur $\widehat{p_{kj}}$ est bien un maximum.

Ainsi :

$$\boxed{\widehat{p_{kj}} = \frac{1}{n_k}\sum_{i=1}^{n} z_{ik}x_{ij}}$$

- L'EMV de π_k :

Soit $k \in [\![1, g]\!]$.

$\widehat{\pi_k}$ maximise la vraisemblance, tout en prenant en compte la contrainte $\sum_{k=1}^{g} \pi_k = 1$.

Tout d'abord, on calcule la dérivée de la vraisemblance par-rapport à π_k :

$$\partial ln(L(\Psi))\partial\pi_k = \partial\partial\pi_k(\sum_{i=1}^{n}\sum_{k=1}^{g}[z_{ik}ln(\pi_k) + \sum_{j=1}^{p} z_{ik}(x_{ij}ln(p_{kj}) + (1-x_{ij})ln(1-p_{kj}))])$$

Or on a fixé k :

$$\Leftrightarrow \partial ln(L(\Psi))\partial\pi_k = \partial\partial\pi_k(\sum_{i=1}^{n} z_{ik}ln(\pi_k))$$

$$\Leftrightarrow \partial ln(L(\Psi))\partial\pi_k = \frac{1}{\pi_k}\sum_{i=1}^{n} z_{ik}$$

Afin de prendre en compte la contrainte $\sum_{k=1}^{g}\pi_k = 1$, on passe par la formulation lagrangienne du problème d'optimisation sous contrainte :

$$\mathcal{L}(L(\Psi),\lambda) = ln(L(\Psi)) - \lambda(\sum_{k=1}^{g}\pi_k - 1)$$

où λ est le multiplicateur de Lagrange associé à la contrainte.

L'application des conditions d'optimalité à ce Lagrangien donnent :

$$\begin{cases} \partial\mathcal{L}(L(\Psi),\lambda)\partial\pi_k = 0 \quad \Leftrightarrow \quad \frac{1}{\pi_k}\sum_{i=1}^{n} z_{ik} = \lambda \quad \Leftrightarrow \quad \pi_k = \frac{1}{\lambda}\sum_{i=1}^{n} z_{ik} \\ \partial\mathcal{L}(L(\Psi),\lambda)\partial\lambda = 0 \quad \Leftrightarrow \quad \sum_{k=1}^{g}\pi_k = 1 \end{cases}$$

On en déduit :

$$\sum_{k=1}^{g}\pi_k = 1 \quad \Leftrightarrow \quad \sum_{k=1}^{g}\frac{1}{\lambda}\sum_{i=1}^{n} z_{ik} = 1 \quad \Leftrightarrow \quad \lambda = \sum_{k=1}^{g}\sum_{i=1}^{n} z_{ik}$$

$$\Leftrightarrow \quad \lambda = \sum_{k=1}^{g} n_k \quad \Leftrightarrow \quad \lambda = n$$

Ce qui nous permet d'écrire :

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

Suivie de cette partie, nous avons développé un modèle spécifique sur le jeu de données *spambase2*, on trouve un taux d'erreur moyen de 11.71

```
$moy
[1] 0.1170515

$var
[1] 4.720492e-05
```

3 Conclusion

Ce projet sur l'apprentissage supervisé nous a permis de mettre en pratique de différentes méthodes de discriminations sur des jeux de données. Nous avons donc pu être confronté aux difficultés de trouver le modèle le plus adapté au jeu de données. En particulier, nos différents résultats souligne l'importance de bien comprendre les hypoyhèses faites par chaque méthode ainsi que d'avoir un bon compromis entre robustesse et flexibilité. Il faut mettre ce dernier en rapport avec la quantité de données disponible.

4 Annexe

4.1 adq.app

```
adq.app <- function(Xapp, zapp)
{
  n <- dim(Xapp)[1]
  p <- dim(Xapp)[2]
  g <- max(unique(zapp))

  param <- NULL
  param$MCov <- array(0, c(p,p,g))
  param$mean <- array(0, c(g,p))
  param$prop <- rep(0, g)

  for (k in 1:g)
  {
    indk <- which(zapp==k)
    #individus de la classe k
    Xk <- Xapp[indk,]
    #nb d'individus dans la classe k
    nk <- length(indk)

    param$MCov[,,k] <- cov(Xk)
    param$mean[k,] <- apply(Xk, 2, mean)
    param$prop[k] <- nk / n
  }

  return(param)
}
```

4.2 adl.app

```r
adl.app <- function(Xapp, zapp)
{
  n <- dim(Xapp)[1]
  p <- dim(Xapp)[2]
  g <- max(unique(zapp))

  param <- NULL
  MCov <- array(0, c(p,p))
  param$MCov <- array(0, c(p,p,g))
  param$mean <- array(0, c(g,p))
  param$prop <- rep(0, g)

  for (k in 1:g)
  {
    indk <- which(zapp==k)
    #individus de la classe k
    Xk <- Xapp[indk,]
    #nb d'individus dans la classe k
    nk <- length(indk)

    MCov <- MCov + nk * cov(Xk)
    param$mean[k,] <- apply(Xk, 2, mean)
    param$prop[k] <- nk / n
  }
  MCov <- MCov / n

  for (k in 1:g)
  {
    param$MCov[,,k] <- MCov
  }

  return(param)
}
```

4.3 cba.app

```r
nba.app <- function(Xapp, zapp)
{
  n <- dim(Xapp)[1]
  p <- dim(Xapp)[2]
  g <- max(unique(zapp))

  param <- NULL

  param$MCov <- array(0, c(p,p,g))
  param$mean <- array(0, c(g,p))
  param$prop <- rep(0, g)

  for (k in 1:g)
  {
    indk <- which(zapp==k)
    #individus de la classe k
    Xk <- Xapp[indk,]
    #nb d'individus dans la classe k
    nk <- length(indk)

    MCov <- array(0, c(p,p))
    diag(MCov) <- diag(cov(Xk))
    param$MCov[,,k] <- MCov
    param$mean[k,] <- apply(Xk, 2, mean)
    param$prop[k] <- nk / n
  }

  return(param)
}
```

4.4 ad.val

```r
ad.val <- function(param, Xtst)
{
  n <- dim(Xtst)[1]
  p <- dim(Xtst)[2]
  g <- length(param$prop)

  out <- NULL

  prob <- matrix(0, nrow=n, ncol=g)

  for (k in 1:g)
  {
    # loi de densité pour la classe k
    fk <- mvdnorm(Xtst, param$mean[k,], param$MCov[,,k])
    #on multiplie par la probabilité à priori pik
    prob[,k] <- param$prop[k] * fk
  }

  prob <- prob / apply(prob,1,sum)
  pred <- max.col(prob)

  out$prob <- prob
  out$pred <- pred

  return(out)
}
```

4.5 log.app

```
log.app <- function(Xapp, zapp, intr, epsi)
{
  n <- dim(Xapp)[1]
  p <- dim(Xapp)[2]

  Xapp <- as.matrix(Xapp)

  if (intr == T)
  {
    Xapp <- cbind(rep(1,n),Xapp)
    p <- p + 1
  }

  targ <- matrix(as.numeric(zapp),nrow=n)
  targ[which(targ==2),] <- 0
  tXap <- t(Xapp)

  beta <- matrix(0,nrow=p,ncol=1)

  conv <- F
  iter <- 0
  while (conv == F)
  {
    iter <- iter + 1
    bold <- beta

    prob <- postprob(beta, Xapp)
    MatW <- diag(as.numeric(prob * (1-prob)))

    beta <- bold + ginv(tXap%*%MatW%*%Xapp) %*% tXap %*% (targ - prob)

    if (norm(beta-bold)<epsi)
    {
      conv <- T
    }
  }

  prob <- postprob(beta, Xapp)
  out <- NULL
  out$beta <- beta
  out$iter <- iter
  out$logL <- sum(targ * log(prob) + (1-targ) * log(1-prob))

  return(out)
}
```

4.6 log.val

```
log.val <- function(beta, Xtst)
{
  m <- dim(Xtst)[1]
  p <- dim(beta)[1]
  pX <- dim(Xtst)[2]

  Xtst <- as.matrix(Xtst)

  if (pX == (p-1))
  {
    Xtst  <- cbind(rep(1,m),Xtst)
  }

  prob_w1 <- postprob(beta, Xtst)
  prob_w2 <- 1 - prob_w1
  prob <- cbind(prob_w1, prob_w2)
  pred <- max.col(prob)

  out <- NULL
  out$prob <- prob
  out$pred <- pred

  return(out)
}

postprob <- function(beta, X)
{
  X <- as.matrix(X)
  prob <- exp(X%*%beta) / (1 + exp(X%*%beta))
}
```