TP2 : Analyse Discriminante PLS exploratoire (ADPLS)

Thomas Anne - laure

2025-11-05

Introduction

L'Analyse Discriminante PLS exploratoire (ADPLS) est une méthode multivariée qui combine les principes de la régression PLS (Partial Least Squares) et de l'analyse discriminante.
Son objectif est de construire des composantes linéaires des variables explicatives X maximisant la capacité de discrimination des groupes définis par la variable qualitative Y.

Autrement dit, on cherche à extraire des axes factoriels $f = XMu$ expliquant à la fois une grande part de la variance de X (inertie totale) et une forte part de la variance expliquée par les classes de Y.
Les notations utilisées sont celles du cours :

- $X \in \mathbb{R}^{n\times p}$: matrice des variables quantitatives centrées ;
- $Y \in \mathbb{R}^{n\times q}$: matrice d'indicatrices des q classes (non centrée) ;
- $W = \mathrm{diag}(w_i)$: matrice diagonale des poids individuels ;
- M : métrique définie positive sur l'espace des variables ;
- $f = XMu$: composante discriminante associée à la direction u ;
- $\Pi_Y = Y(Y'WY)^{-1}Y'W$: projecteur sur l'espace engendré par Y ;
- $\widehat{X} = \Pi_Y X$: projection de X sur cet espace.

Partie 1

1. Première identité : $\|f\|_W^2 R^2(f,Y) = \|\widehat{X}Mu\|_W^2$

Par définition, la proportion de variance de la composante f expliquée par Y est donnée par :

$$R^2(f,Y) = \frac{\|\Pi_Y f\|_W^2}{\|f\|_W^2}.$$

Or,

$$\Pi_Y f = \Pi_Y(XMu) = (\Pi_Y X)Mu = \widehat{X}Mu.$$

Ainsi, on obtient immédiatement :

$$\boxed{\|f\|_W^2 R^2(f,Y) = \|\widehat{X}Mu\|_W^2.}$$

Cette identité exprime que le produit de la norme pondérée de f par sa qualité de prédiction $R^2(f,Y)$ correspond à la norme pondérée de la projection de X sur l'espace des classes, transformée par Mu.
Elle constitue le point de départ du critère d'optimisation de l'ADPLS.

2. Programme de rang 1

On considère la matrice :

$$E = \widehat{X}'W\widehat{X}.$$

2-a) Interprétation de E

La matrice E représente la **matrice d'inertie inter-classes** ou la **covariance des centres de gravité des classes** projetés.
Si m_k désigne le centre de gravité de la classe k et m la moyenne globale, on peut écrire :

$$E = \sum_k n_k (m_k - m)(m_k - m)'.$$

Les valeurs propres de E mesurent la **dispersion inter-classes** dans les différentes directions, et permettent d'identifier les axes de plus grande discrimination.

2-b) Programme d'optimisation

On cherche le vecteur u maximisant la part de variance expliquée par les classes :

$$\max_{u'Mu=1} \|\widehat{X}Mu\|_W^2 = \max_{u'Mu=1} u'(MEM)u.$$

C'est un **problème de Rayleigh**, dont la condition d'optimalité conduit à :

$$EMu = \eta u,$$

où u est le **vecteur propre M-unitaire** associé à la plus grande valeur propre η.
La première composante discriminante s'écrit alors :

$$f_1 = XMu_1.$$

2-c) Symétrisation du problème

On introduit :

$$u^* = M^{1/2}u, \qquad X^* = XM^{1/2},$$

d'où $\widehat{X}^* = \Pi_Y X^*$.

Ainsi,

$$E^* = (\widehat{X}^*)'W\widehat{X}^* = M^{1/2}EM^{1/2}.$$

L'équation propre devient :

$$E^* u^* = \eta u^*.$$

La matrice E^* est **symétrique définie positive**, et u^* est un vecteur propre euclidien associé à la plus grande valeur propre η.
La première composante peut donc s'écrire :

$$\boxed{f_1 = X^* u_1^*.}$$

3. Programme de rang h

À partir de la deuxième composante, on impose la contrainte d'orthogonalité :

$$F'_{h-1} W f_h = 0,$$

où $F_{h-1} = [f_1, \ldots, f_{h-1}]$.

On pose également :

$$D' = F'_{h-1} W X.$$

Le problème d'optimisation devient alors :

$$\max_{u'Mu=1,\, D'Mu=0} u'MEMu.$$

3-a) Équation propre restreinte

L'écriture du lagrangien et les conditions d'optimalité conduisent à l'équation :

$$\Pi_{D^\perp} EMu = \lambda u,$$

avec

$$\Pi_{D^\perp} = I - D(D'MD)^{-1} D'M,$$

qui est le **projecteur sur le sous-espace orthogonal à** D pour la métrique M.

3-b) Sous-espace orthogonal

Puisque $u \in \langle D^\perp \rangle$, on peut écrire l'équation sous la forme :

$$\Pi_{D^\perp} EM \Pi_{D^\perp} u = \lambda u.$$

3-c) Forme symétrique

En posant $u^* = M^{1/2} u$, on obtient l'équation propre symétrique :

$$\boxed{M^{1/2} \Pi_{D^\perp} \widehat{X}' W \widehat{X} \Pi'_{D^\perp} M^{1/2} u^* = \lambda u^*.}$$

Cette formulation assure que la matrice à diagonaliser est **symétrique**, ce qui garantit l'orthogonalité des composantes extraites.

4. Indicateurs de qualité des composantes

Pour chaque composante $f = XMu$, on définit :

$$S(f) = \frac{\|f\|_W^2}{\mathrm{tr}(X'WX)}, \qquad R^2(f, Y) = \frac{\|\widehat{X}Mu\|_W^2}{\|f\|_W^2}.$$

- $S(f)$ mesure la **part de l'inertie totale** de X expliquée par la composante f.
- $R^2(f, Y)$ mesure la **part de la variance de** f expliquée par les classes Y.
- Le produit $S(f)\, R^2(f, Y)$ représente le **pouvoir discriminant** de la composante.

Ces indicateurs permettent d'évaluer la qualité globale du modèle ADPLS et l'importance relative de chaque axe discriminant.

5. Représentations graphiques

Les résultats de l'ADPLS peuvent être visualisés sous deux formes complémentaires :
le **plan des individus et des centres de gravité des classes**, et le **plan des variables**.

5-a) Centres de gravité des classes

Les coordonnées des q centres de gravité des classes sur les H premiers axes discriminants sont données par :

$$\boxed{(Y'WY)^{-1}Y'W\widetilde{F}_H,}$$

où $\widetilde{F}_H = [\tilde{f}_1, \ldots, \tilde{f}_H]$ contient les composantes réduites.
Ces points résument la position moyenne de chaque classe dans l'espace discriminant.

5-b) Représentation des variables

Chaque variable x_j peut être représentée dans le plan (h, m) par ses corrélations avec les composantes :

$$\left(\frac{\langle x_j, f_h \rangle_W}{\|x_j\|_W \, \|f_h\|_W}, \; \frac{\langle x_j, f_m \rangle_W}{\|x_j\|_W \, \|f_m\|_W} \right).$$

Ces coordonnées permettent d'interpréter la signification de chaque axe discriminant en fonction des variables les plus corrélées.
Le **cercle unité** sert de repère visuel pour évaluer la qualité de représentation des variables dans le plan factoriel.

Partie 2

```r
# ------------------------------------------------------------------------
# Fonctions utilitaires pondérées
# ------------------------------------------------------------------------

# création matrice diagonale W à partir d'un vecteur de poids w
Wmat <- function(w) {
  if (is.null(w)) return(NULL)
  Diagonal <- diag(as.numeric(w), nrow = length(w))
  return(Diagonal)
}

# norme pondérée ||v||_W^2 = v' W v avec w vecteur poids ou W matrice diag
w_norm2 <- function(v, w) {
  if (is.null(w)) return(sum(v^2))
  return(as.numeric(t(v) %*% (w * v)))
}

# produit scalaire pondéré <a,b>_W = a' W b
w_prod <- function(a, b, w) {
  if (is.null(w)) return(as.numeric(t(a) %*% b))
  return(as.numeric(t(a) %*% (w * b)))
}

# racine carrée symétrique de M (M supposée SPD)
M_half <- function(M) {
  ev <- eigen(M, symmetric = TRUE)
  vals <- ev$values
  vecs <- ev$vectors
  vals[vals < 0] <- 0
  sqrtM <- vecs %*% diag(sqrt(vals), length(vals)) %*% t(vecs)
  return(sqrtM)
}

# inverse racine
M_half_inv <- function(M) {
  ev <- eigen(M, symmetric = TRUE)
  vals <- ev$values
  vecs <- ev$vectors
  vals[vals < 0] <- 0
  invsqrt <- vecs %*% diag(ifelse(vals > 0, 1/sqrt(vals), 0), length(vals)) %*% t(vecs)
  return(invsqrt)
}

# ------------------------------------------------------------------------
# Fonction principale : adpls
# ------------------------------------------------------------------------
adpls <- function(X, Y, H = 2, w = NULL, M = NULL, center_scale = TRUE) {
  n <- nrow(X); p <- ncol(X)
  if (is.null(w)) w <- rep(1/n, n)
  w <- as.numeric(w)
  if (length(w) != n) stop("Longueur de w différente de nrow(X)")
  W <- diag(w)
```

```r
if (center_scale) {
  X <- scale(X, center = TRUE, scale = TRUE)
} else {
  X <- as.matrix(X)
}
Y <- as.matrix(Y)

if (is.null(M)) M <- diag(1, p)
if (!all(dim(M) == c(p, p))) stop("M doit être p x p")

sqrtM <- M_half(M)
inv_sqrtM <- M_half_inv(M)

# projecteur Pi_Y
YYw <- t(Y) %*% (W %*% Y)
inv_YYw <- solve(YYw)
PiY <- Y %*% inv_YYw %*% t(Y) %*% W

Xhat <- PiY %*% X
E <- t(Xhat) %*% (W %*% Xhat)

# stockage
U_list <- list()
F <- matrix(0, n, H)
Ftilde <- matrix(0, n, H)
S_vec <- numeric(H)
R2_vec <- numeric(H)
var_coords <- matrix(0, p, H)

for (h in 1:H) {
  if (h == 1) {
    Xstar <- X %*% sqrtM
    Xhat_star <- PiY %*% Xstar
    Estar <- t(Xhat_star) %*% (W %*% Xhat_star)
    ev <- eigen(Estar, symmetric = TRUE)
    ustar <- ev$vectors[, 1]
    u <- inv_sqrtM %*% ustar
    norm_u <- as.numeric(t(u) %*% (M %*% u))
    if (norm_u <= 0) stop("Norme non positive détectée pour u")
    u <- u / sqrt(norm_u)
  } else {
    Fprev <- F[, 1:(h-1), drop = FALSE]
    Dprime <- t(Fprev) %*% (W %*% X)
    D <- t(Dprime)
    temp <- solve(t(D) %*% (M %*% D))
    PiDperp <- diag(1, p) - D %*% temp %*% t(D) %*% M
    Estar_proj <- sqrtM %*% PiDperp %*% E %*% t(PiDperp) %*% sqrtM
    ev <- eigen(Estar_proj, symmetric = TRUE)
    ustar <- ev$vectors[, 1]
    u <- inv_sqrtM %*% ustar
    norm_u <- as.numeric(t(u) %*% (M %*% u))
    if (norm_u <= 0) stop("Norme non positive détectée pour u (h>1)")
    u <- u / sqrt(norm_u)
```

```
  }

  # scores
  f <- X %*% (M %*% u)
  fnorm2 <- as.numeric(t(f) %*% (W %*% f))
  if (fnorm2 <= 0) {
    warning(paste("Composante", h, "a norme nulle ou négative (arrêt)."))
    break
  }
  ftilde <- f / sqrt(fnorm2)

  # indicateurs
  total_inertia <- sum(diag(t(X) %*% (W %*% X)))
  S_val <- fnorm2 / total_inertia
  num <- Xhat %*% (M %*% u)
  top_val <- as.numeric(t(num) %*% (W %*% num))
  R2_val <- top_val / fnorm2

  # corrélations variables
  var_corrs <- numeric(p)
  denom_f <- sqrt(fnorm2)
  for (j in 1:p) {
    denom_xj <- sqrt(as.numeric(t(X[, j]) %*% (W %*% X[, j])))
    if (denom_xj == 0) {
      var_corrs[j] <- 0
    } else {
      num_j <- as.numeric(t(X[, j]) %*% (W %*% f))
      var_corrs[j] <- num_j / (denom_xj * denom_f)
    }
  }

  # stockage
  U_list[[h]] <- u
  F[, h] <- as.numeric(f)
  Ftilde[, h] <- as.numeric(ftilde)
  S_vec[h] <- S_val
  R2_vec[h] <- R2_val
  var_coords[, h] <- var_corrs
}

# centres de gravité
centers <- solve(t(Y) %*% (W %*% Y)) %*% t(Y) %*% (W %*% Ftilde)
rownames(centers) <- if (!is.null(colnames(Y))) colnames(Y) else paste0("class", 1:ncol(Y))
colnames(centers) <- paste0("ax", 1:ncol(Ftilde))

result <- list(U = do.call(cbind, U_list),
               F = F,
               Ftilde = Ftilde,
               S = S_vec,
               R2 = R2_vec,
               centers = centers,
               var_coords = var_coords,
               Xhat = Xhat,
```

```
                  E = E,
                  M = M,
                  w = w)
  return(result)
}

# ------------------------------------------------------------------------
# Fonction de tracé : plot_adpls
# ------------------------------------------------------------------------
plot_adpls <- function(out, Y = NULL, Y_labels = NULL, h = 1, m = 2,
                       show_variables = TRUE, circle = TRUE,
                       main = NULL, cex_ind = 0.7, cex_cent = 1.0) {
  Ftilde <- out$Ftilde
  centers <- out$centers
  var_coords <- out$var_coords
  n <- nrow(Ftilde)
  if (is.null(Y) && is.null(Y_labels)) stop("Donner Y (indicatrices) ou Y_labels (facteur).")
  if (is.null(Y_labels)) {
    Ymat <- as.matrix(Y)
    lab <- apply(Ymat, 1, function(r) which(r == 1))
    Y_labels <- factor(lab)
  }
  xind <- Ftilde[, h]; yind <- Ftilde[, m]
  classes <- as.factor(Y_labels)
  cols <- rainbow(length(levels(classes)))
  plot(xind, yind, col = cols[as.numeric(classes)], pch = 16, cex = cex_ind,
       xlab = paste0("Axe ", h), ylab = paste0("Axe ", m), main = main)
  legend("topright", legend = levels(classes), col = cols, pch = 16, cex = 0.8)
  points(centers[, h], centers[, m], col = "black", pch = 8, cex = cex_cent)
  text(centers[, h], centers[, m], labels = rownames(centers), pos = 3)
  if (show_variables) {
    vx <- var_coords[, h]; vy <- var_coords[, m]
    arrows(rep(0, length(vx)), rep(0, length(vy)), vx, vy, length = 0.07, col = "darkgrey")
    text(vx, vy, labels = colnames(out$M) %||% paste0("x", 1:nrow(var_coords)), cex = 0.7)
    if (circle) {
      angles <- seq(0, 2 * pi, length.out = 200)
      lines(cos(angles), sin(angles), lty = 2)
    }
  }
}

# infix helper
`%||%` <- function(a, b) if (!is.null(a)) a else b
```

Exemple d'application avec les données `Datagenus` du premier TP :

```
data <- read.table("Datagenus.csv", header = TRUE)

X <- scale(as.matrix(data[, paste0("gen", 1:27)]), center = TRUE, scale = TRUE)
grp <- as.factor(data$forest)
Y <- model.matrix(~ grp - 1)

out <- adpls(X, Y, H = 3, w = NULL, M = NULL, center_scale = FALSE)
```

```r
# Tableau indicateurs
data.frame(
  Axe = 1:3,
  S = round(out$S, 3),
  R2 = round(out$R2, 3),
  Discriminant = round(out$S * out$R2, 3)
)
```

```
##   Axe     S    R2 Discriminant
## 1   1 0.184 0.200         0.037
## 2   2 0.116 0.178         0.021
## 3   3 0.051 0.090         0.005
```

```r
# Graphique plan 1-2
plot_adpls(out, Y = Y, Y_labels = grp, h = 1, m = 2, show_variables = TRUE)
```