METODOS PLS EN ANALISIS MULTIBLOQUES: TRATAMIENTO DE DATOS FALTANTES Y MULTICOLINEALIDAD

PhD. VICTOR MANUEL GONZALEZ ROJAS
UNIVERSIDAD DEL VALLE – ESCUELA DE ESTADISTICA
UNIVERSIDAD POLITECNICA DE CATALUÑA

SANTIAGO DE CALI, ABRIL 2016

EL ALGORITMO NIPALS

(NONLINEAR ESTIMATION BY ITERATIVE PARTIAL LEAST SQUARE)

EL ALGORITMO NIPALS

El algoritmo NIPALS (Nonlinear estimation by Iterative Partial Least Square) es la base de la regresión PLS, Wold (1966). Fundamentalmente realiza una descomposición singular (SVD) de una matriz de datos, mediante secuencias iterativas de proyecciones ortogonales [concepto geométrico de regresión]. Con bases de datos completas se tiene equivalencia con los resultados del ACP; sin embargo, y esta quizá es su mayor virtud, se puede realizar el ACP con datos faltantes (missing data) y obtener sus estimaciones a partir de la matriz de datos reconstituida.

Sea $X_{n,p}$ la matriz de datos de rango *a* cuyas columnas $X_1,..,X_p$ se suponen centradas. La descomposición derivada del ACP conlleva a:

$$X = \sum_h^a t_h P_h^{'}$$ [*t* es la *compal* (*scores*) y $P_h^{'}$ el *vect_p* (*loadings*) en el eje *h*]

$$\left[X_1 \ldots X_p\right] = t_1 P_1^{'} + \ldots + t_a P_a^{'}. \qquad [1]$$

Así, la columna $X_j = \sum_h^a p_{hj} t_h \quad j = 1,..p$ y la i-ésima fila $x_i = \sum_h^a t_{hi} P_h \quad i = 1, n.$

Observe entonces que si *h=1*, la columna *j* se expresa como $X_j = p_{1j} t_1$ es decir $p_{hj} = X_j^{'} t_h$ es como el coeficiente (pendiente) en la regresión de X_j sobre t_h. En el espacio de las filas, t_{hi} es el coeficiente de la regresión sin constante del individuo x_i sobre P_h.

Para h>1, p_{hj} es el coeficiente de regresión de t_h en la regresión simple del vector deflactado $X_j - \sum_l^{h-1} p_{lj}\ t_l$ sobre t_h y t_{hi} el de P_h en la regresión de $x_i - \sum_l^{h-1} t_{li}\ P_l$ sobre P_h .

El objetivo de cualquier algoritmo PLS es el procedimiento iterativo para calcular los *parámetros* del modelo. Para cada componente las cargas son computadas, una como función de la otra, a través del procedimiento iterativo.

Descripción pseudocódigo NIPALS.

El flujograma asociado al procedimiento de convergencia en la etapa 2.2, es:

$$X = X_0 \longrightarrow t_1 \longrightarrow P_1^+ = X' t_1 / t_1' t_1 \longrightarrow P_1 = \frac{P_1^+}{\|P_1^+\|}$$

$$t_1 = X P_1 / P_1' P_1$$

Se construirán una serie de tablas notadas $\mathbf{X}_h$ cuyas columnas son $X_{h1},\ldots,X_{hp}$; la i-ésima fila se notará $x'_{hi} = (x_{h1i}, \ldots, x_{hpi})$. El algoritmo inicia tomando X_{01} como la 1ª *compal* t_1.

A. Sin datos faltantes:

Etapa 1. $X_o = X_h$

Etapa 2. *h*=1,2, …, *a:*

Etapa 2.1. t_h = 1ª columna de X_{h-1} [prop $\bar{X}$]

Etapa 2.2. : repetir hasta la convergencia de P_h

Etapa 2.2.1 $P_h = \frac{X'_{h-1} t_h}{t'_h t_h}$ $\left[u = \frac{X' X u}{\lambda} \ , \ \lambda = \frac{1}{n-1} t' t\right]$

Etapa 2.2.2 normar p_h a 1

Etapa 2.2.3 $t_h = X_{h-1} P_h / P'_h P_h$ [t = Xu]

Etapa2.3 $X_h = X_{h-1} - t_h P'_h$ [garantiza la ortogonalidad]

Siguiente h.

Tal como se estudio inicialmente, en la etapa 2.2.1 p_{hj} *representa*, antes de la normalización, el coeficiente [pendiente] de la regresión de $X_{h-1,j}$ sobre la componente t_h. En la etapa 2.2.3 t_{hi} es el largo de la proyección ortogonal de $x_{h-1,i}$ sobre P_h .

Para $h=1$ se obtiene el primer eje factorial P_1 y la primera *compal* t_1 de $X'X$. Ya que la matriz $X_1 = X - t_1 P_1^{'}$ representa el *residuo* de la regresión de X sobre la primera *compal*, de [1], el vect_p P_2 de la matriz X'_1X_1 /(n-1) asociado al *val_p* más grande, corresponde al *vect_p* de $X'X/(n-1)$ asociado al segundo *val_p* más grande λ_2 .

Una vez se consigue la convergencia, en la etapa 2.3 se deflacta la matriz precedente para garantizar la ortogonalidad de las siguientes componentes.

Así, el problema del ACP es resolver una serie de regresiones simples locales hasta alcanzar la convergencia de los coeficientes de regresión p_{hj} y t_{hi} que es el nuevo valor proporcionado de la regresión sin constante de $x_{h-1,i}$ sobre la 'nueva' variable P_h después de la normalización.

Si hay datos faltantes se obtiene sin embargo las componentes t_h y los vectores P_h que permiten luego 'reconstituir' la matriz X y de ésta, estimar los datos faltantes.

La principal característica del NIPALS es que trabaja respecto a una serie de productos escalares como suma de productos de los elementos emparejados. Esto permite manejar missing data, agregando en cada operación los pares disponibles. Geométricamente el procedimiento 'toma' los elementos missing como si ellos cayeran sobre la recta de regresión; no son puntos de apalancamiento.

B. Pseudocodigo NIPALS con datos faltantes

Etapa 1. $X_o = X_h$

Etapa 2. *h*=1,2, …, *a:*

Etapa 2.1. t_h = 1ª columna de X_{h-1}

Etapa 2.2. : repetir hasta la convergencia de P_h

Etapa 2.2.1. Para j=1,2,…,p:

$$p_{hj} = \frac{\Sigma_{\{i:x_{ji}\ e\ t_{hi}\ existen\}} x_{h-1,ji}\, t_{hi}}{\{i:x_{ji}\ e\ t_{hi}\ existen\}\, t_{hi}^2}$$

[$\mathrm{cov}(t_h, x_{h-1,j})/s^2_{th}$]

Etapa 2.2.2. Normar P_h a 1.

Etapa 2.2.3 Para i=1,2,…,n:

$$t_{hi} = \frac{\Sigma_{\{j:x_{ji}\ existe\}} x_{h-1,ji}\, p_{hj}}{\Sigma_{\{j:x_{ji}\ existe\}} p_{hj}^2}$$

Etapa 2.3 $X_h = X_{h-1} - t_h P_h^{'}$

End

- En las etapas 2.2.1 y 2.2.3 se calculan las pendientes de las rectas de mínimos cuadrados pasando por el origen de la nube de puntos sobre los *datos disponibles* . Los P_{hj} y los t_{hi} deben conservar en sus posiciones *j* e *i*, la característica de dato faltante dada por x_{ij}, la cual se puede expresar con 0.

Así, el algoritmo NIPALS permite estimar los datos faltantes utilizando la formula de reconstitución habitual derivada del ACP : $\hat{x}_{ji} = \sum_{l}^{h} t_{li} p_{lj}$.

ALGORITMO NIPALS bajo R [vmg]

Base de datos cuantitativa completa [sin NA]

```
# Crear el archivo troue.xls y guardarlo como .txt delimitado
# por tabulaciones. Abra R y cambie a Directorio de trabajo
# conteniendo el .txt

     Xi <- read.table("troue.txt",header=TRUE)
     Xii <- Xi[,-1]
     rownames(Xii) <- Xi[,1]
     ------------------------------------------------------------
     library(ade4)
     data(doubs)
     Xii <- doubs$mil
     X <- sqrt(n/(n-1))*scale(Xii)
     ------------------------------------------------------------

     n <- nrow(Xii)
     p <- ncol(X)
     X0 <- X
     T <- matrix(1,n,p)
     P <- matrix(1,p,p)
```

```
for(h in 1:p)
     {
        t1 <- as.matrix(X0[,1])

        for(e in 1:20)
        {
           P11 <- (t(X0)%*%t1)/(as.numeric(t(t1)%*%t1))
           nP11 <- as.numeric(t(P11)%*%P11)
           P1 <- 1/sqrt(nP11)*P11    # matrix
           t1 <- X0%*%P1
        }

        T[,h] <- t1
        P[,h] <- P1
        X1 <- X0 - t1%*%t(P1)   # deflacta
        X0 <- X1
     }
     t1 <- T[,1]; t2 <- T[,2]; ...
     p1 <- P[,1]; p2 <- P[,2]; ... u1=P1 , u=P
```

```
# NIPALS entrega las compals T y los vect-p P tal como con ade4,
# equivalentes, excepto tal vez por signo. Puede usar svd().

     acpDoubs <- dudi.pca(Xii,scale=T, scannf=F, nf=5)

     t1 = acpDoubs$l1 = X%*%as.matrix(acpDoubs$c1[1]) = (Xu)
     t(t1)%*%t1/n = 1ºval_p = 6.321624.


# NOTA : con datos completos parece no ser necesaria la division
             t(t1)%*%t1 en P11.


# End, Nipals.   troue Bajo R
```

NIPALS - NA

(MISSING DATA)

Algoritmo bajo R

```
fnipNA <- function(Xi)            # X : Xi  base de datos con y sin datos faltantes
{
    library(far)
    p <- ncol(Xi); n <- nrow(Xi)

      njm <- colMeans(Xi,na.rm=TRUE); njs <- colSums(Xi,na.rm=TRUE)
      nj <- njs/njm   # vector conteniendo los n sin NA en c|colj

    Xo <- scale(Xi)*sqrt(nj/(nj-1))

    P <- matrix(0,p,p); T <- matrix(0,n,p)
    P1i <- matrix(0,p,1)

    for(h in 1:p)
    {
       t1 <- Xo[,1]

       for(e in 1:100)
       {
                for(j in 1:p)
                {
                  j1 <- na.omit(cbind(Xo[,j],t1))
                  P1i[j] <- sum(j1[,1]*j1[,2])/sum(j1[,2]^2)
                }
                P[,h] <- P1i
                Portn <- orthonormalization(P[,1:h]); P1 <- Portn[,h]
```

```
        for(i in 1:n)
        {
          i1 <- na.omit(cbind(Xo[i,],P1))
          t1[i] <- sum(i1[,1]*i1[,2])/sum(i1[,2]^2)
        }
        T[,h]<- t1
        Tortg <- orthonormalization(T[,1:h],norm=FALSE);  t1 <- Tortg[,h]

      }  # end e

      P[,h] <- P1
      T[,h] <- t1

      X1 <- Xo - t1%*%t(P1); Xo <- X1

    }  # end h

    L <- diag(t(T)%*%T)/n
    r.nipNA <- list(T,P,L); return(r.nipNA)

}   # end  fnipNA, conserva máxima inercia eje por eje
```

```
X; XE

          CILIN     PUISS     VITES     POIDS      LONG     LARGE
hoci         NA -0.610090 -0.320110 -1.101723 -1.231959 -0.326790
re19  -0.362936        NA -0.078518 -0.619522 -0.156625  0.081697
fiati -0.669918 -0.803769        NA -0.598557 -0.624161  0.217860
p405  -0.258431 -0.610090 -0.078518        NA  0.427796  0.081697
re21   0.392545 -0.665427 -0.078518  0.093295        NA  0.217860
cibx  -0.258431 -0.610090  0.002013 -0.221183  0.053767        NA
bm530        NA  2.101420  1.773693  1.665687  1.175855  0.898672
ro827  1.714092        NA  1.612631  1.057696  1.105724  0.898672
re25   1.437590  1.935409        NA  0.994800  1.152478  1.579484
opom   0.240143  0.275301  0.324137        NA  1.199232  1.170997
p405b  0.037665  0.358307  0.485199  0.030400        NA  0.354022
fosi   0.229257  0.081622  0.122810  0.323913  0.684941        NA
bm325        NA  1.631056  1.048915  0.785148  0.240781 -0.599115
au90   0.231434        NA  1.290507  0.449704  0.404419  0.081697
fosc   2.275804  1.050018        NA  0.973835  1.035594  1.034834
rees   0.233611  0.219964 -0.199314        NA  0.334289  1.170997
niva   0.139993 -0.693095 -1.528073  1.330243        NA  0.081697
vwca   0.481809 -0.001383 -1.326746  0.869009  0.825202        NA
fofi         NA -1.716828 -1.890462 -1.269444 -1.185205 -0.871439
fiat1 -1.680128        NA -1.487808 -1.395236 -1.348843 -1.824576
p205  -0.669918 -0.886774        NA -0.975931 -1.208582 -1.688414
p205r -1.292590 -0.250400  0.283871        NA -1.208582 -1.552252
seati -0.929002 -0.333405 -0.038252 -0.787244        NA -1.007602
ciax  -1.292590 -0.471747  0.082544 -1.604888 -1.676119        NA
```

XE

	CILIN	PUISS	VITES	POIDS	LONG	LARGE
hoci	-1.302486	-0.610090	-0.320110	-1.101723	-1.231959	-0.326790
re19	-0.362936	-0.378871	-0.078518	-0.619522	-0.156625	0.081697
fiati	-0.669918	-0.803769	-0.678750	-0.598557	-0.624161	0.217860
p405	-0.258431	-0.610090	-0.078518	0.029052	0.427796	0.081697
re21	0.392545	-0.665427	-0.078518	0.093295	0.363316	0.217860
cibx	-0.258431	-0.610090	0.002013	-0.221183	0.053767	-0.038869
bm530	1.986146	2.101420	1.773693	1.665687	1.175855	0.898672
ro827	1.714092	1.399367	1.612631	1.057696	1.105724	0.898672
re25	1.437590	1.935409	1.407520	0.994800	1.152478	1.579484
opom	0.240143	0.275301	0.324137	0.686175	1.199232	1.170997
p405b	0.037665	0.358307	0.485199	0.030400	0.107128	0.354022
fosi	0.229257	0.081622	0.122810	0.323913	0.684941	0.491779
bm325	1.029555	1.631056	1.048915	0.785148	0.240781	-0.599115
au90	0.231434	0.628551	1.290507	0.449704	0.404419	0.081697
fosc	2.275804	1.050018	1.169634	0.973835	1.035594	1.034834
rees	0.233611	0.219964	-0.199314	0.331437	0.334289	1.170997
niva	0.139993	-0.693095	-1.528073	1.330243	0.806327	0.081697
vwca	0.481809	-0.001383	-1.326746	0.869009	0.825202	0.948670
fofi	-1.717985	-1.716828	-1.890462	-1.269444	-1.185205	-0.871439
fiat1	-1.680128	-1.617983	-1.487808	-1.395236	-1.348843	-1.824576
p205	-0.669918	-0.886774	-0.601973	-0.975931	-1.208582	-1.688414
p205r	-1.292590	-0.250400	0.283871	-1.286662	-1.208582	-1.552252
seati	-0.929002	-0.333405	-0.038252	-0.787244	-0.848864	-1.007602
ciax	-1.292590	-0.471747	0.082544	-1.604888	-1.676119	-1.655527

Los valores sombreados son estimaciones de la matriz X estandarizada [bajo n-NA^{s}]; ahora, cuales son las estimaciones originales ??

Gracias por su atencion!!