Algebra lineal R4

Juan Luis Acebal Rico

1. Prediccion meteorologica En la television publica de vuestro paıs quieren relevar al meteorologo de cabecera para calcular y presentar la prediccion meteorologica en prime time. Despues de un duro proceso de seleccion os acaban seleccionando y hoy es el primer dıa de trabajo. Como especialistas en ciencia de datos, lo primero que quereis hacer es analizar perıodos historicos temporales para observar los diferentes patrones y ver si se replican en el tiempo. Estais interesados en conocer cuales han sido las variables mas relevantes para la prediccion a lo largo del tiempo. Para realizar esta tarea, os proporcionan las observaciones diarias (a las 12 del mediodıa) de diferentes variables relacionadas con la meteorologıa durante un perıodo de 3 anos (2006-2008). weather label : el tiempo del dıa (nublado: 0, lluvioso: 1, soleado: 2). temperature: temperatura (en grados centıgrados). temp app: sensacion termica (en grados centıgrados). humidity: humedad relativa (en tanto por uno [0-1]). wind_vel : velocidad del viento (en kilometros por hora). wind_dir : direccion del viento (en grados). visibility: visibilidad (en kilometros). atm_pres: presion atmosf´erica (en milibares). Una libreria que os puede ser util para realizar la practica es fields, como vereis mas adelante. Recordar que debeis instalarla una sola vez y luego importarla en el codigo. Antes de empezar, debeis abrir la “Tabla resumen de la Practica 1” del Moodle. Allı, encontrareis el valor de los parametros (T , E) para poder realizar la practica. Recordar, tambien, que debeis indicar los valores utilizados al inicio de la memoria, ası como el intento de la Tabla correspondiente (primero o segundo).

#Pregunta 0. Los resultados del informe corresponden al segundo intento(T=2 y E=80)

```
if (!require(fields)) {
  install.packages("fields")
}
```

```
## Loading required package: fields
```

```
## Loading required package: spam
```

```
## Spam version 2.11-0 (2024-10-03) is loaded.
## Type 'help( Spam)' or 'demo( spam)' for a short introduction
## and overview of this package.
## Help for individual functions is also obtained by adding the
## suffix '.spam' to the function name, e.g. 'help( chol.spam)'.
```

```
##
## Attaching package: 'spam'
```

```
## The following objects are masked from 'package:base':
##
##     backsolve, forwardsolve
```

```
## Loading required package: viridisLite
```

```
##
## Try help(fields) to get started.
```

```
library(fields)

datos0608 <- read.csv('PR1_T2024_data/data_0608.csv')
```

```r
datos0911 <- read.csv('PR1_T2024_data/data_0911.csv')
E <- 80
T <- 2
```

```r
head(datos0608)
```

```
##   weather_label temperature temp_app humidity wind_vel wind_dir visibility
## 1             1    16.06111 16.06111     0.58   4.7817      212     9.9015
## 2             0    13.81667 13.81667     0.31  20.7690      282    11.4954
## 3             0    12.28889 12.28889     0.42   7.8890      236     9.9820
## 4             0    21.11111 21.11111     0.38  23.3289      299     9.9820
## 5             0    12.17222 12.17222     0.32  30.5739      329    11.2056
## 6             0    10.88889 10.88889     0.58  23.2967      319    11.2700
##   atm_pres
## 1  1024.37
## 2  1017.78
## 3  1020.96
## 4  1009.09
## 5     0.00
## 6  1010.76
```

```r
head(datos0911)
```

```
##   weather_label temperature temp_app humidity wind_vel wind_dir visibility
## 1             0    17.24444 17.24444     0.41  14.8281       10    16.1000
## 2             0    13.79444 13.79444     0.43  17.8710      319    15.8263
## 3             0    13.81667 13.81667     0.51  20.0123       10    10.2557
## 4             0    14.97222 14.97222     0.42   4.4436      183     9.9820
## 5             0    17.22222 17.22222     0.38   7.7119      315    10.2557
## 6             0    17.86111 17.86111     0.37  17.5812      241    10.2557
##   atm_pres
## 1  1012.66
## 2  1016.34
## 3  1018.05
## 4  1017.68
## 5  1016.18
## 6  1013.75
```

```r
etiquetas <- datos0608$weather_label
caracteristicas <- datos0608[, -1]
```

```r
head(etiquetas)
```

```
## [1] 1 0 0 0 0 0
```

```r
head(caracteristicas)
```

```
##   temperature temp_app humidity wind_vel wind_dir visibility atm_pres
## 1    16.06111 16.06111     0.58   4.7817      212     9.9015  1024.37
## 2    13.81667 13.81667     0.31  20.7690      282    11.4954  1017.78
## 3    12.28889 12.28889     0.42   7.8890      236     9.9820  1020.96
## 4    21.11111 21.11111     0.38  23.3289      299     9.9820  1009.09
## 5    12.17222 12.17222     0.32  30.5739      329    11.2056     0.00
## 6    10.88889 10.88889     0.58  23.2967      319    11.2700  1010.76
```

```r
dim(datos0608)
```

```
## [1] 1096    8
```

```r
dim(datos0911)
```

```
## [1] 1096    8
```

#Pregunta 1 [10 %] Primeramente, leer el fichero de datos correspondiente al perıodo 2006-2008. 1 > data_df <- read . csv ('/home / data_0608. csv ') De la tabla resultante, guardar la primer columna (weather label) al vector y y las otras columnas a la matriz de caracterısticas X. Responder: ¿que dimension tiene la matriz X?

```r
dim(caracteristicas) # 1096 filas y 7 columnas
```

```
## [1] 1096    7
```

#Pregunta 2 [10 %] Antes de realizar cualquier tipo de analisis, es importante hacer una exploracion estadıstica (cuantitativa y cualitativa) de los datos. Para este proposito, observar el numero de dıas con tiempo T y calcular su temperatura media (solo de los dıas correspondientes a T)

```r
sapply(sort(unique(etiquetas)), function(label) {
  mean(caracteristicas[etiquetas == label, "temperature"], na.rm = TRUE)
})
```

```
## [1] 16.505284  6.036585 22.933632
```

```r
dias_T1 <- sum(etiquetas == T)  #dias con T=1
temp_media_T1 <- mean(caracteristicas[etiquetas == T, "temperature"], na.rm = TRUE)
```

```r
cat("Días con T = 1:", dias_T1, "\n")  # 164
```

```
## Días con T = 1: 93
```

```r
cat("Temperatura media T = 1:", round(temp_media_T1, 2), "°C\n")  # 6.04C°
```

```
## Temperatura media T = 1: 22.93 °C
```

#Pregunta 3 [10 %] Para poder aplicar la descomposicion en componentes principales, debeis normalizar la matriz de datos X siguiendo los criterios de la Seccion 2.1 de los apuntes del modulo. Para hacerlo, debeis calcular la media y la desviacion tıpica de los datos; guardar ambas en las variables m_X y s_X, respectivamente, ya que las necesitareis mas adelante. Nombrar a la nueva matriz de datos normalizada Xs. Una vez hecho, indicar la temperatura media de todo el perıodo 2006-2008 de los datos normalizados.

```r
# media y std de las columnas de caracteristicas
media_caracteristicas <- colMeans(caracteristicas, na.rm = TRUE)
desviacion_caracteristicas <- apply(caracteristicas, 2, sd, na.rm = TRUE)

caracteristicas_normalizadas <- scale(caracteristicas, center = media_caracteristicas, scale = desviaci

temp_media_normalizada <- mean(caracteristicas_normalizadas[, "temperature"], na.rm = TRUE)
```

```r
#caracteristicas_normalizadas
```

```r
cat("Temperatura media normalizada:", temp_media_normalizada, "\n")
```

```
## Temperatura media normalizada: -7.323825e-16
```

Es una temperatura cercana a cero que es correcto para un dato normalizado, donde la media tendria que dar cero.

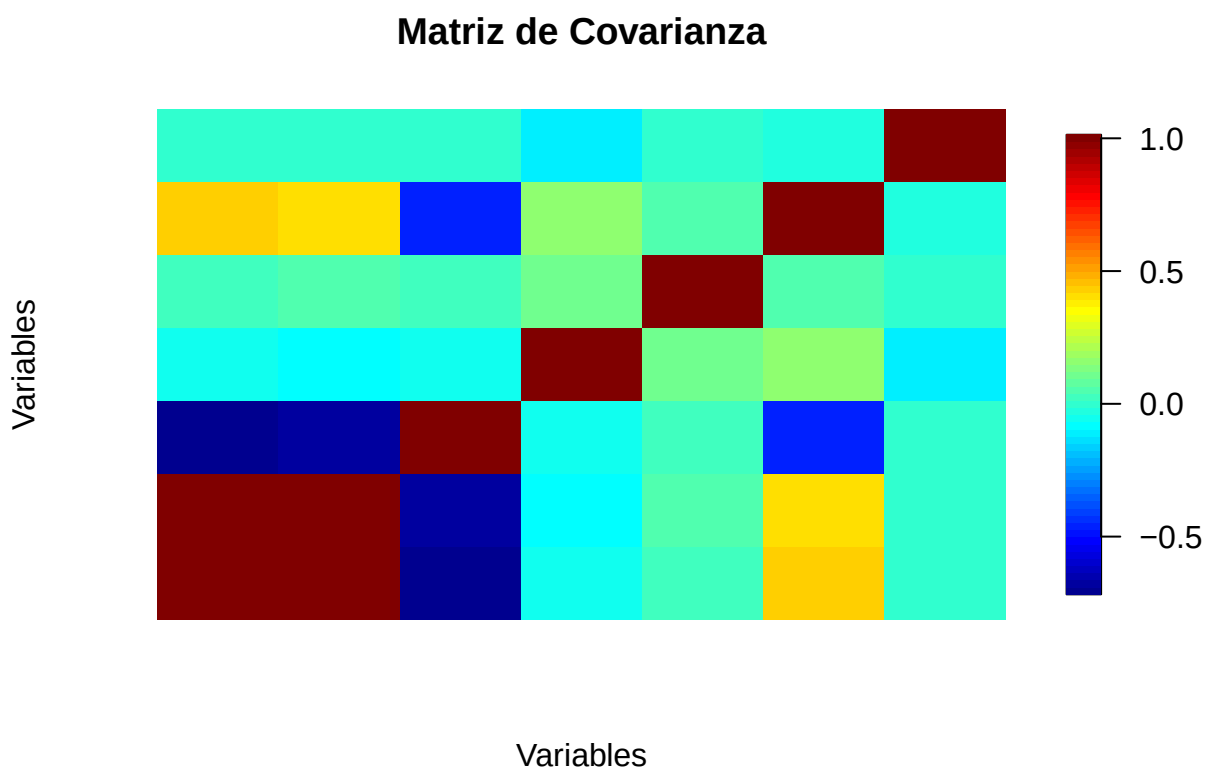
#Pregunta 4 [15 %] Para ver la relacion cruzada entre las distintas variables, observar la matriz de covarianza CXs. Dibujarla mediante la instruccion image.plot() de la librerıa fields y contestar: • ¿Que variable esta mas asociada a la visibilidad en valor absoluto (y sin que sea ella misma)? • ¿Que variable esta menos asociada a la visibilidad en valor absoluto?

```r
CXs <- cov(caracteristicas_normalizadas) # covarianza: CXs
asociaciones <- CXs["visibility", -match("visibility", colnames(CXs))]
variable_maxima <- names(asociaciones)[which.max(abs(asociaciones))]
variable_minima <- names(asociaciones)[which.min(abs(asociaciones))]
```

```r
CXs
```

```
##                temperature     temp_app     humidity    wind_vel     wind_dir
## temperature   1.000000000  0.994038596 -0.704059749 -0.04984219 0.0345141425
## temp_app      0.994038596  1.000000000 -0.684842263 -0.08653176 0.0430674352
## humidity     -0.704059749 -0.684842263  1.000000000 -0.04809928 0.0265759712
## wind_vel     -0.049842192 -0.086531757 -0.048099282  1.00000000 0.1080245214
## wind_dir      0.034514142  0.043067435  0.026575971  0.10802452 1.0000000000
## visibility    0.420755056  0.413125571 -0.449830633  0.16559802 0.0471598933
## atm_pres     -0.000735414  0.006641105 -0.009984373 -0.10134570 0.0005196018
##                visibility      atm_pres
## temperature   0.42075506 -0.0007354140
## temp_app      0.41312557  0.0066411049
## humidity     -0.44983063 -0.0099843725
## wind_vel      0.16559802 -0.1013456994
## wind_dir      0.04715989  0.0005196018
## visibility    1.00000000 -0.0253288165
## atm_pres     -0.02532882  1.0000000000
```

```r
image.plot(CXs, main = "Matriz de Covarianza", xlab = "Variables", ylab = "Variables", axes=FALSE)
```



```r
asociaciones
```

```
## temperature    temp_app    humidity    wind_vel    wind_dir    atm_pres
##  0.42075506  0.41312557 -0.44983063  0.16559802  0.04715989 -0.02532882
```

```r
cat("Variable con mayor covarianza:", variable_maxima, "(", max(abs(asociaciones)), ")\n")  # humidity
```

```
## Variable con mayor covarianza: humidity ( 0.4498306 )
```

```r
cat("Variable con menor covarianza:", variable_minima, "(", min(abs(asociaciones)), ")\n")  # atm_pres
```

```
## Variable con menor covarianza: atm_pres ( 0.02532882 )
```

La variable con mayor covarianza con visibility es humidity con 0.44983063 y la que menor atm_pres con 0.0253. Ambas me refiero en valor absoluto.

Por ultimo, tambien vemos que temperature y temp_app comparten mucha varianza y humidity con temperature, temp_app y visibility

#Pregunta 5 [15 %] Seguidamente, calcular la descomposici´on en componentes principales de la matriz de covarianza CXs. Utilizar la instruccion eigen y consultar la documentaci´on si lo necesitais(1). Esta funcion proporciona los valores y vectores propios (componentes principales) de forma ordenada de mayor a menor varianza explicada de los datos originales. Ası, la primera componente corresponde a la direccion de maxima varianza mientras que la ultima componente corresponde a la direccion de mınima varianza. Dibujar la distribucion de la varianza acumulada (eje de ordenadas) para cada componente principal (eje de abscisas) respecto a la varianza total de los datos. Seguidamente, indicar el numero mınimo de componentes necesarios P para explicar un E % de la varianza inicial de los datos.

```r
pca <- eigen(CXs)
varianza_acumulada <- cumsum(pca$values) / sum(pca$values)
P <- min(which(varianza_acumulada >= E / 100))
```

```r
cat("PCAs necesarias para explicar", E, "% de la varianza:", P, "\n")
```

```
## PCAs necesarias para explicar 80 % de la varianza: 4
```

#Pregunta 6 [10 %] Con el tiempo, se os encarga un nuevo estudio, ahora durante el perıodo 2009-2011. Como suposicion inicial, considerar una distribucion estacionaria de los datos, eso es, que sus propiedades estadısticas son constantes en el tiempo. Esto os permite utilizar la media y desviacion tıpica anteriormente calculadas ası como las componentes principales del perıodo 2006-2008.

Empezar leyendo los datos del nuevo perıodo y normalizarlos usando la media y desviacion tıpica previamente calculadas (apartado 3); guardar los datos normalizados a la matriz Xs_test. Una vez hecho, contestar: ¿cuantos dıas de T habeis observado en este segundo perıodo?

```r
etiquetas_0911 <- datos0911$weather_label
dias_T_0911 <- sum(etiquetas_0911 == T)

caracteristicas_0911_normalizadas <- scale(datos0911[, -1], center = media_caracteristicas, scale = des
```

```r
cat("Días con T =", T, "en el segundo período:", dias_T_0911, "\n")
```

```
## Días con T = 2 en el segundo período: 56
```

#Pregunta 7 [15 %] Utilizando las P primeras componentes principales calculadas anteriormente, proyectar los datos del perıodo 2009-2011 (normalizados) al nuevo subespacio. Guardar dicha proyeccion a la variable Xproj_test. Recordar que este subespacio tiene dimension P.

Responder: ¿que proporcion (en porcentaje) de la varianza inicial de los datos explican los datos del subespacio, Xproj_test? ¿Es mayor o menor que la obtenida en el perıodo 2006-2008?

```r
var_total_test <- sum(apply(caracteristicas_0911_normalizadas, 2, var))
Xproj_test <- caracteristicas_0911_normalizadas %*% pca$vectors[, 1:P]
var_subespacio_test <- sum(apply(Xproj_test, 2, var))
varianza_explicada_test <- var_subespacio_test / var_total_test * 100
cat("Porcentaje real de varianza explicada:", round(varianza_explicada_test,3))
```

```
## Porcentaje real de varianza explicada: 81.459
```

```r
#var_total_test
```

#Pregunta 8 [15 %] Finalmente, a partir de la proyeccion Xproj_test quereis recuperar los datos observados tal y como se indica a la Seccion 2.5 y 2.5.1 de los apuntes del modulo. Calcular el error de reconstruccion y responder: ¿cual es la desviacion tıpica del error de reconstruccion de la temperatura? ¿Creeis que la suposicion de una distribucion estacionaria de los datos es correcta?

```r
proyeccion_0911 <- caracteristicas_0911_normalizadas %*% pca$vectors[, 1:P]
reconstruccion <- proyeccion_0911 %*% t(pca$vectors[, 1:P])
```

```r
#head(reconstruccion)
```

```r
reconstruccion_desnormalizada <- sweep(reconstruccion, 2, desviacion_caracteristicas, "*")
reconstruccion_desnormalizada <- sweep(reconstruccion_desnormalizada, 2, media_caracteristicas, "+")
```

```r
#head(reconstruccion_desnormalizada)
```

```r
error_reconstruccion_desnormalizado <- reconstruccion_desnormalizada - datos0911[, -1]
desviacion_error <- sd(error_reconstruccion_desnormalizado[, "temperature"])
```

```r
#head(error_reconstruccion_desnormalizado)
```

```r
cat("std (temperature):", round(desviacion_error, 2), "\n")
```

```
## std (temperature): 3.29
```

```r
std_temp_0608 <- sd(datos0608$temperature, na.rm = TRUE)
std_temp_0911 <- sd(datos0911$temperature, na.rm = TRUE)
datos_combinados <- rbind(datos0608, datos0911)
std_temp_combinada <- sd(datos_combinados$temperature, na.rm = TRUE)
```

```r
cat("std(temperature) (2006-2011):", std_temp_combinada, "\n")
```

```
## std(temperature) (2006-2011): 9.916005
```

```r
cat("std(temperature) (2006-2008):", std_temp_0608, "\n")
```

```
## std(temperature) (2006-2008): 10.66402
```

```r
cat("std(temperature) (2009-2011):", std_temp_0911, "\n")
```

```
## std(temperature) (2009-2011): 9.110014
```

No parece correcta, o al menos, no se puede utilizar el primer periodo para predecir el segundo facilmente, sin embargo, si comparamos el std del primer periodo, con el segundo, y el proyectado, vemos que es el que tiene una desviacion estandar mas baja. Esto indica que las PCA funcionan bien, es decir capturan la informacion importante, pero NO para predecir el segundo periodo, es decir, o bien las temperaturas no son constantes, es decir ha cambiado el clima, o algun tipo de ajuste.