Sesión 3: Componentes principales y Newton Raphson

Aplicaciones en Computación Estadística

Natalie Julian - www.nataliejulian.com

Estadística UC y Data Scientist en Zippedi Inc.

Ejercicio propuesto sesión anterior

Ejercicio propuesto

- Cree una función que utilice SWEEP y que entregue los coeficientes estimados de una regresión lineal, el valor estimado de R^2, errores estándar, estadísticos y valores p del test t para cada coeficiente.
- Compare con la función `lm()`. ¿Cuáles son las variables más significativas? ¿cómo es el ajuste del modelo? Comente.

Una solución

```
library(ISR3)
library(dplyr)

regresion <- function(x,y){ #x: matriz de predictores y:variable respuesta
  W <- cbind(1,x,y)
  Z <- crossprod(W)
  p <- dim(Z)[1]-1
  gl.e <- length(y)-p
  gl.m <- p-1
  Z <- SWP(Z,1:p)
  SCT <- crossprod(y-mean(y))
  SCE <- Z[p+1, p+1]
  SCM <- SCT-SCE
  CME <- SCE/gl.e
  CMM <- SCM/gl.m
  R2 <- as.numeric(SCM/SCT)
  beta<-data.frame(Z[1:p,p+1], sqrt(diag(-CME*(Z[1:p,1:p]))))
  names(beta)<-c("Estimate", "std.error")
  test<-beta %>%
        mutate(t=Estimate/std.error, pvalue=2*(1-pt(abs(t),gl.e)))
  coeficientes <- round(test,4)
  row.names(coeficientes)<-c("Intercept", colnames(x))
  list(R.squared = R2, Coef = coeficientes)
  }
```

Comparación con función `lm()`

```
regresion(x, happyness)$Coef
                Estimate std.error       t pvalue
Intercept        -0.3133    0.1555 -2.0151 0.0453
acousticness      0.2163    0.0682  3.1697 0.0018
danceability      0.7481    0.0986  7.5848 0.0000
energy            0.5492    0.1248  4.4017 0.0000
key              -0.0004    0.0034 -0.1180 0.9062
liveness         -0.0724    0.1282 -0.5644 0.5732
loudness          0.0146    0.0102  1.4376 0.1522
speechiness      -0.1500    0.1364 -1.0997 0.2729
instrumentalness -0.7029    0.5988 -1.1737 0.2420

summary(lm(happyness~x))$coefficients
                      Estimate  Std. Error    t value     Pr(>|t|)
(Intercept)       -0.3133110829 0.155482872 -2.0150842 4.532023e-02
xacousticness      0.2162959293 0.068238763  3.1696930 1.781747e-03
xdanceability      0.7480721321 0.098627312  7.5848375 1.490260e-12
xenergy            0.5491542059 0.124758701  4.4017307 1.798161e-05
xkey              -0.0003958872 0.003355673 -0.1179755 9.062130e-01
xliveness         -0.0723786190 0.128238736 -0.5644053 5.731511e-01
xloudness          0.0146291490 0.010176064  1.4376039 1.522088e-01
xspeechiness      -0.1500250674 0.136419664 -1.0997320 2.728551e-01
xinstrumentalness -0.7028724995 0.598829611 -1.1737437 2.419825e-01

regresion(x, happyness)$R.squared
[1] 0.3816236

summary(lm(happyness~x))$r.squared
[1] 0.3816236
```

Componentes principales

Componentes principales

Considere el ejercicio anterior, donde existía cierto grado de multicolinealidad entre las variables predictoras. En dichos casos es necesario reducir la dimensionalidad. Cuando hablamos de reducir dimensionalidad, hablamos de considerar una matriz de diseño con menor número de columnas.

Componentes principales

El análisis de componentes principales es vastamente utilizado para efectos de reducción de dimensionalidad. Sobretodo cuando tenemos problemas de multicolinealidad pero queremos utilizar toda la información obtenida, se trata de utilizando las p variables, determinar p vectores ortogonales a través de combinaciones lineales de estas p covariables. La idea es que, al ser vectores ortogonales esto resulta ser conveniente computacionalmente y podemos elegir k con $k \leq p$ componentes principales de manera de reducir dimensionalidad o aprovechar la propiedad de ortogonalidad.

Ejercicio 1

Continuando con el ejercicio de `spotify`:

a) Realice análisis de componentes principales. Comente sobre la variabilidad explicada con las componentes, realice un gráfico adecuado.

b) Comente sobre la magnitud de las variables en las primeras dos componentes principales. ¿Cuáles serían las variables con mayor y menor contribución en cada una?. Entregue un gráfico para observar la influencia de las variables en todas las componentes. Comente.

c) Plantee distintos criterios en los que podría basar su elección del número de componentes principales a elegir.

d) ¿Qué se puede hacer con las componentes principales una vez halladas? Comente.

Ejercicio 2

Instacart es una empresa que ofrece servicios de delivery de alimentos en los Estados Unidos y Canadá. Los usuarios seleccionan los productos del despacho a través de su sitio web o de la aplicación móvil. La información de compras y entregas de Instacart se encuentra en las bases de datos `order products train`, `orders`, `products`.

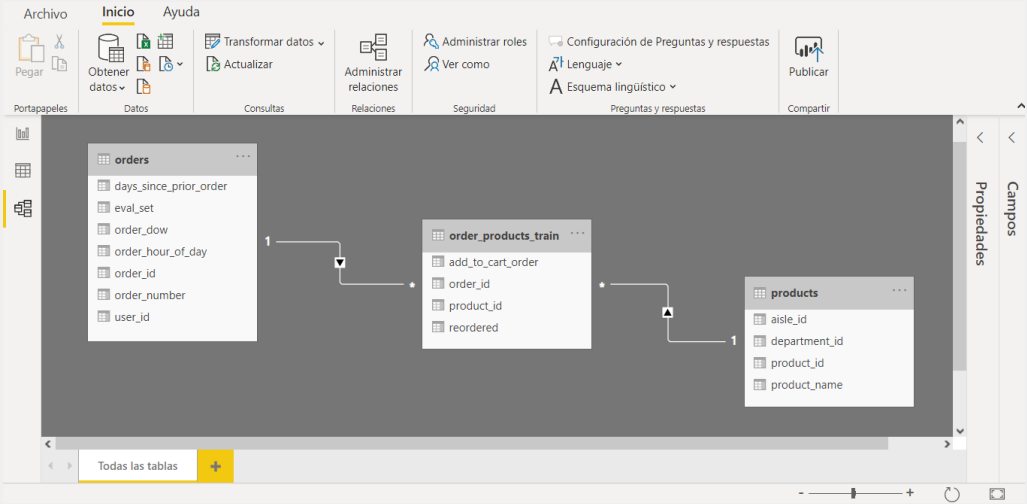
Tablas de datos

Las tablas de datos entregan la siguiente información:

- `products` Contiene información de los productos, nombre de los productos e ID de los productos
- `order_products_train` Contiene por orden (pedido) los productos despachados
- `orders` Entrega detalles de la orden (pedido), id del cliente y detalles logísticos de la entrega

La información se estructura en un modelo relacional.

Modelo relacional en PowerBI



Ejercicio 2

Instacart quiere detectar patrones de compra de sus clientes, por lo que le interesa modelar por cliente y pedido, la proporción de productos que ya habían sido pedidos para delivery anteriormente. La variable *reordered* indica con un 1 si el producto ha sido pedido más de una vez para delivery y 0 si no.

Ejercicio 2

a) Realice el cruce correspondiente de las tablas, de manera de, en una tabla de datos, poseer la información por cliente, pedido y producto.

b) Por cliente y pedido obtenga la proporción de productos que ya habían sido pedidos anteriormente el cliente.

c) Para el análisis, no resulta interesante analizar aquellos pedidos en los que todos los productos ya habían sido pedidos anteriormente y tampoco aquellos en los que ninguno había sido pedido anteriormente. Elimine estos casos. Se cree que *prop* constituye una muestra aleatoria de **X**, con $x_i \sim Beta(\alpha, \beta)$. Sin embargo, sus parámetros se desconocen. Utilice Newton Raphson de manera de obtener estimaciones para α y β.

Métodos de optimización

El algoritmo de Newton Raphson es:

$$\theta^{(t+1)} = \theta^{(t)} - (\frac{d^2 l(\theta)}{d\theta d\theta^t})^{-1} \bigtriangledown l(\theta)$$

El algoritmo de Fisher-Scoring es:

$$\theta^{(t+1)} = \theta^{(t)} + (I(\theta))^{-1} \bigtriangledown l(\theta)$$

Con $I(\theta) = -E(\frac{d^2 l(\theta)}{d\theta d\theta^t})$ (en la familia exponencial)

Algoritmo de Newton-Raphson

i) Obtener verosimilitud de los parámetros dado theta.

ii) Obtener la log-verosimilitud de los parámetros dado theta.

iii) A partir de la log-verosimilitud obtener el vector gradiente $\triangledown l(\theta)$ (vector de derivadas parciales de la log-verosimilitud respecto a cada parámetro) y obtener la matriz hessiana (matriz de las segundas derivadas parciales).

iv) Obtener la matriz hessiana $H(\theta) = \frac{d^2 l(\theta)}{d\theta d\theta^t}$, si no posee nada en función de las variables aleatorias, entonces es una matriz constante (luego, el algoritmo de Newton Raphson y Fisher Scoring coinciden.) En ese caso, el algoritmo puede escribirse de manera simplificada:

$$\theta^{(t+1)} = \theta^{(t)} - (\frac{d^2 l(\theta)}{d\theta d\theta^t})^{-1} \triangledown l(\theta)$$

Newton Raphson

$$L(\alpha, \beta|x) = \prod_{i=1}^{n} \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x_i^{\alpha-1}(1-x_i)^{\beta-1}$$ Obtener verosimilitud de parámetros condicionado a la muestra

Newton Raphson

$$L(\alpha,\beta|x) = \prod_{i=1}^{n} \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x_i^{\alpha-1}(1-x_i)^{\beta-1}$$ Obtener verosimilitud de parámetros condicionada a la muestra

$$= \left(\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\right)^{n} \left(\prod_{i=1}^{n} x_i\right)^{\alpha-1} \left(\prod_{i=1}^{n} 1-x_i\right)^{\beta-1}$$ Escribir de manera conveniente para aplicarle logaritmo

Newton Raphson

$$L(\alpha,\beta|x) = \prod_{i=1}^{n} \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x_i^{\alpha-1}(1-x_i)^{\beta-1}$$ Obtener verosimilitud de parámetros condicionada a la muestra

$$= \left(\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\right)^n \left(\prod_{i=1}^{n} x_i\right)^{\alpha-1} \left(\prod_{i=1}^{n}(1-x_i)\right)^{\beta-1}$$ Escribir de manera conveniente para aplicarle logaritmo

Obtenemos la log-verosimilitud:

$$l(\alpha,\beta|x) = n\left[\log(\Gamma(\alpha+\beta)) - \log(\Gamma(\alpha)) - \log(\Gamma(\beta)\right] + (\alpha-1)\sum_{i=1}^{n}\log(x_i) + (\beta-1)\sum_{i=1}^{n}\log(1-x_i)$$

Newton Raphson

$$L(\alpha,\beta|x) = \prod_{i=1}^{n} \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x_i^{\alpha-1}(1-x_i)^{\beta-1}$$ Obtener verosimilitud de parámetros condicionada a la muestra

$$= \left(\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\right)^n \left(\prod_{i=1}^{n} x_i\right)^{\alpha-1} \left(\prod_{i=1}^{n}(1-x_i)\right)^{\beta-1}$$ Escribir de manera conveniente para aplicarle logaritmo

Obtenemos la log-verosimilitud:

$$l(\alpha,\beta|x) = n\left[\log(\Gamma(\alpha+\beta)) - \log(\Gamma(\alpha)) - \log(\Gamma(\beta)\right] + (\alpha-1)\sum_{i=1}^{n}\log(x_i) + (\beta-1)\sum_{i=1}^{n}\log(1-x_i)$$

Luego las derivadas parciales respecto a α y β:

$$\nabla l(\theta) = \left(n\frac{\Gamma'(\alpha+\beta)}{\Gamma(\alpha+\beta)} - n\frac{\Gamma'(\alpha)}{\Gamma(\alpha)} + \sum_{i=1}^{n}\log(x_i)\ ,\ n\frac{\Gamma'(\alpha+\beta)}{\Gamma(\alpha+\beta)} - n\frac{\Gamma'(\beta)}{\Gamma(\beta)} + \sum_{i=1}^{n}\log(1-x_i)\right)$$

Newton Raphson

$$L(\alpha,\beta|x) = \prod_{i=1}^{n} \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x_i^{\alpha-1}(1-x_i)^{\beta-1} \quad \text{Obtener verosimilitud de parámetros condicionada a la muestra}$$

$$= \left(\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\right)^n \left(\prod_{i=1}^{n} x_i\right)^{\alpha-1} \left(\prod_{i=1}^{n}(1-x_i)\right)^{\beta-1} \quad \text{Escribir de manera conveniente para aplicarle logaritmo}$$

Obtenemos la log-verosimilitud:

$$l(\alpha,\beta|x) = n\left[\log(\Gamma(\alpha+\beta)) - \log(\Gamma(\alpha)) - \log(\Gamma(\beta)\right] + (\alpha-1)\sum_{i=1}^{n}\log(x_i) + (\beta-1)\sum_{i=1}^{n}\log(1-x_i)$$

Luego, obtenemos la derivada de la log-verosimilitud respecto a α y β respectivamente:

$$\nabla l(\theta) = \left(n \underbrace{\frac{\Gamma'(\alpha+\beta)}{\Gamma(\alpha+\beta)}}_{\texttt{digamma(alpha+beta)}} - n\frac{\Gamma'(\alpha)}{\Gamma(\alpha)} + \sum_{i=1}^{n}\log(x_i) \;,\; n\frac{\Gamma'(\alpha+\beta)}{\Gamma(\alpha+\beta)} - n\frac{\Gamma'(\beta)}{\Gamma(\beta)} + \sum_{i=1}^{n}\log(1-x_i) \right)$$

digamma(x): En R, entrega la derivada de $\log(\Gamma(x))$.

Newton Raphson

Finalmente, la matriz de derivadas parciales, con θ el vector de parámetros:

$$\frac{d^2 l(\theta)}{d\theta d\theta^t} = n \begin{bmatrix} \frac{\Gamma''(\alpha+\beta)\Gamma(\alpha+\beta)-(\Gamma'(\alpha+\beta))^2}{(\Gamma(\alpha+\beta))^2} - \frac{\Gamma''(\alpha)\Gamma(\alpha)-(\Gamma'(\alpha))^2}{(\Gamma(\alpha))^2} & \frac{\Gamma''(\alpha+\beta)\Gamma(\alpha+\beta)-(\Gamma'(\alpha+\beta))^2}{(\Gamma(\alpha+\beta))^2} \\ \frac{\Gamma''(\alpha+\beta)\Gamma(\alpha+\beta)-(\Gamma'(\alpha+\beta))^2}{(\Gamma(\alpha+\beta))^2} & \frac{\Gamma''(\alpha+\beta)\Gamma(\alpha+\beta)-(\Gamma'(\alpha+\beta))^2}{(\Gamma(\alpha+\beta))^2} - \frac{\Gamma''(\beta)\Gamma(\beta)-(\Gamma'(\beta))^2}{(\Gamma(\beta))^2} \end{bmatrix}$$

Newton Raphson

Finalmente, la matriz de derivadas parciales, con θ el vector de parámetros:

$$n\begin{bmatrix} \frac{\Gamma''(\alpha+\beta)\Gamma(\alpha+\beta)-(\Gamma'(\alpha+\beta))^2}{(\Gamma(\alpha+\beta))^2} - \frac{\Gamma''(\alpha)\Gamma(\alpha)-(\Gamma'(\alpha))^2}{(\Gamma(\alpha))^2} & \frac{\Gamma''(\alpha+\beta)\Gamma(\alpha+\beta)-(\Gamma'(\alpha+\beta))^2}{(\Gamma(\alpha+\beta))^2} \\ \frac{\Gamma''(\alpha+\beta)\Gamma(\alpha+\beta)-(\Gamma'(\alpha+\beta))^2}{(\Gamma(\alpha+\beta))^2} & \frac{\Gamma''(\alpha+\beta)\Gamma(\alpha+\beta)-(\Gamma'(\alpha+\beta))^2}{(\Gamma(\alpha+\beta))^2} - \underbrace{\frac{\Gamma''(\beta)\Gamma(\beta)-(\Gamma'(\beta))^2}{(\Gamma(\beta))^2}}_{\texttt{trigamma(beta)}} \end{bmatrix}$$

trigamma(x): En R, entrega la segunda derivada de $\log(\Gamma(x))$.

Algoritmo

Luego el algoritmo de Newton Raphson es:

$$\theta^{(t+1)} = \theta^{(t)} - (\frac{dl(\theta)}{d\theta d\theta^t})^{-1} \bigtriangledown l(\theta)$$

Es fácil ver que los algoritmos de Newton Raphson y Score coinciden, puesto que al calcular $I(\theta) = -E(\frac{d^2 l(\theta)}{d\theta d\theta^t})$ obtenemos $I(\theta) = -\frac{d^2 l(\theta)}{d\theta d\theta^t}$ pues la matriz hessiana resultante es una matriz constante respecto a x.

En R

```
beta.NR<-function(theta0, x, error, i=0){
  alpha.old<-theta0[1]
  beta.old<-theta0[2]
  n<-length(x)
  sumlogx<-sum(log(x))
  sumlog1x<-sum(log(1-x))

  grad.l<-c(n*digamma(alpha.old+beta.old)-n*digamma(alpha.old)+sumlogx,
            n*digamma(alpha.old+beta.old)-n*digamma(beta.old)+sumlog1x)

  hess.l<-n*matrix(c(trigamma(alpha.old+beta.old)-trigamma(alpha.old),
                     trigamma(alpha.old+beta.old),trigamma(alpha.old+beta.old),
                     trigamma(alpha.old+beta.old)-trigamma(beta.old)),2,2)
  theta.new<-theta0-solve(hess.l)%*%grad.l

  Rerror<-sqrt(sum((theta.new-theta0)^{2}))

  if(Rerror>error){
    beta.NR(theta.new,x, error, i=i+1)
  }

  else{
    list(alphaybeta=theta.new,"iteraciones"=i,hess.l=hess.l,
         estandarerror=c(sqrt(-solve(hess.l)[1,1]),sqrt(-solve(hess.l))[2,2]))
  }
}
```

En R

```
beta.NR(c(0.1,0.1),x,error=0.01)
$alphaxbeta
         [,1]
[1,] 2.155885
[2,] 1.724135

$iteraciones
[1] 8

$hess.l
          [,1]      [,2]
[1,] -28901.17  28909.63
[2,]  28909.63 -47744.64

$estandarerror
[1] 0.009367406 0.007288106


library(Rfast)

beta.mle(x)
$iters
[1] 5

$loglik
[1] 12603.74

$param
   alpha     beta
2.155885 1.724135
```