Guia 25 Clase

Agustin Muñoz González

15/7/2020

Preparamos el entorno.

```
rm(list=ls())
library(ggplot2)
library(tidyr)
library(gganimate)
datos=read.csv('lidar.csv',header=TRUE)[,1:2]
names(datos)=c('range','logratio')
```

Un poco de contexto.

Tenemos Y: Variable Respuesta, X: Variable Explicativa y queremos g(X) un predictor. Vimos la noción de **error cuadrático medio** $\mathbb{E}[\{Y - g(X)\}^2]$. El mejor predictor es $r(X)$ tal que

$$\mathbb{E}[\{Y - r(X)\}^2] \leq \mathbb{E}[\{Y - g(X)\}^2], \quad \forall g : \mathbb{R}^p \longrightarrow \mathbb{R}.$$

La suposición es entonces pensar que $Y = r(X) + \epsilon$ con X independiente de ϵ y $\mathbb{E}(\epsilon) = 0$.

Pero un problema común es la disponibilidad de los datos, tenemos muchos o pocos datos?

Si tenemos una gran muestra $(X_1, Y_1), \ldots, (X_n, Y_n)$ iid, $(X_i, Y_i) \sim \mathcal{P}$ vimos los estimadores

- Nadaraya-Watson: ventana h;
- Vecinos próximos (knn): vecinos k.

¿Pero qué hacemos si no tenemos tantos datos? Proponemos la regresión lineal/polinomial simple/múltiple. La difencia entre las funciones de regresión propuestas en cada caso radica en la dimensión del vector de parámetros $\boldsymbol{\beta}$ (lineal/polinomial) y en la cantidad de variables explicativas (simple/múltiple). La idea es siempre la misma, hallar el β que minimiza el error cuadrático medio, i.e. usar el método de **cuadrados mínimos** A saber:

- Regresión lineal simple: Asumimos $r(x) = \beta_0^* + \beta_1^* x$. Entonces

$$(\hat{\beta}_0, \hat{\beta}_1) = arg \min_{\beta_0, \beta_1} \sum_{i=1}^{n} \{Y_i - (\beta_0 + \beta_1 X_i)\}^2.$$

  Y haremos las predicciones con $\hat{r}(x) = \hat{\beta}_0 + \hat{\beta}_1 x$;

- Regresión polinomial simple: Asumimos $r(x) = \beta_0^* + \beta_1^* x + \cdots + \beta_p^* x^p$. Entonces

$$(\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_p) = arg \min_{\boldsymbol{\beta}} \sum_{i=1}^{n} \{Y_i - (\beta_0 + \beta_1 X_i + \cdots + \beta_p X_i^p)\}^2.$$

  Y haremos las predicciones con $\hat{r}(x) = \hat{\beta}_0 + \hat{\beta}_1 x + \cdots + \hat{\beta}_p x^p$.

- Regresión lineal multiple: Asumimos $r(x) = \beta_0^* + \beta_1^* x_1 + \cdots + \beta_p^* x_p$. Entonces si notamos $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)$,

$$(\hat{\beta}_0, \hat{\boldsymbol{\beta}}) = arg\min_{\beta_0, \boldsymbol{\beta}} \sum_{i=1}^{n} \{Y_i - (\beta_0 + \boldsymbol{\beta}^t X_i)\}^2.$$

  Y haremos las predicciones con $\hat{r}(x) = \hat{\beta}_0 + \hat{\boldsymbol{\beta}}^t x$.

EL TERMINO β_0 SE LLAMA EL INTERCEPT (JUSTAMENTE PORQUE CUANDO X=0, ES LA INTERSECCION CON EL Y).

Por supuesto el predictor que propongamos podría no ser el $r(X)$, pero en todo caso habremos encontrado el mejor predictor de una cierto conjunto de predictores $\mathcal{F} = \{r_\theta : \theta \in \Theta \subset \mathbb{R}^k\}$.

Veamos un poco el modelo lineal tradicional. Sean Y_i: respuestas, $X_i = (X_{1i}, \ldots, X_{pi})'$: covariables o variables explicativas. Asumimos $Y_i = X_i' \boldsymbol{\beta}^* + \epsilon_i$, es decir, $r(x) = X_i' \boldsymbol{\beta}^*$. Entonces el estimador de **mínimos cuadrados** es

$$\hat{\boldsymbol{\beta}} = arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} \{Y_i - (X_i' \boldsymbol{\beta})\}^2.$$

Derivando e igualando a 0 tenemos que $\hat{\boldsymbol{\beta}}$ es solución del sistema

$$\sum_{i=1}^{n} (Y_i - X_i' \boldsymbol{\beta}) X_i = 0 \iff X'X\boldsymbol{\beta} = X'Y.$$

Estas ultimas expresiones se llaman **Ecuación Normal**. Si XX' es no singular entonces de las ecuaciones normales tenemos

$$\hat{\boldsymbol{\beta}} = (X'X)^{-1} X'Y.$$

Supongamos por último que $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$, entonces $Y_i = X_i' \boldsymbol{\beta}^* + \epsilon_i \sim \mathcal{N}(X_i' \boldsymbol{\beta}^*, \sigma^2)$. Entonces podemos estimar $\boldsymbol{\beta}^*$ usando el método de máxima verosimilitud

$$L(\boldsymbol{\beta}) = L(\boldsymbol{\beta}, (y_1, X_1), \ldots, (y_n, X_n)) = \frac{1}{(\sigma\sqrt{2\pi})^n} e^{(-\frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - X_i' \boldsymbol{\beta})^2)}.$$

Donde

$$\log L(\boldsymbol{\beta}) \approx -\sum_{i=1}^{n} (y_i - X_i' \boldsymbol{\beta})^2,$$

que se maximiza en el mínimo (por el signo -) de

$$\sum_{i=1}^{n} (y_i - X_i' \boldsymbol{\beta})^2.$$

i.e. Si las respuestas ϵ_i tienen distribución normal el estimador de mínimos cuadrados, $\hat{\boldsymbol{\beta}}$, coincide con el estimador de máxima verosimilitud y $\hat{\boldsymbol{\beta}}$ hereda la distribución normal.

Ejercicio I. Continuamos con el ejercicio de los datos de LIDAR de la práctica anterior. La técnica conocida como LIDAR (light detection and ranging) usa la reflexión de luz de láser emitida para detectar compuestos químicos en la atmósfera. Esta técnica ha probado ser una herramienta muy eficiente para el monitoreo de la distribución de diversos elementos polulantes en la atmósfera (Sigrist, 1994). En el archivo lidar.txt se encuentran datos medidos con a la técnica LIDAR. La variable **range** es la distancia recorrida antes de que la luz sea reflejada de regreso hacia su fuente. La variable **logratio** es el logaritmo del cociente de la luz recibida de dos fuentes de luz láser de distinta frecuencia. El objetivo de esta guía es comparar diferentes métodos de predicción, evaluando en un test set. Antes de empezar, separe un 10% de los datos para conformar el **test set** que se empleará en la comparación final.

Preliminar

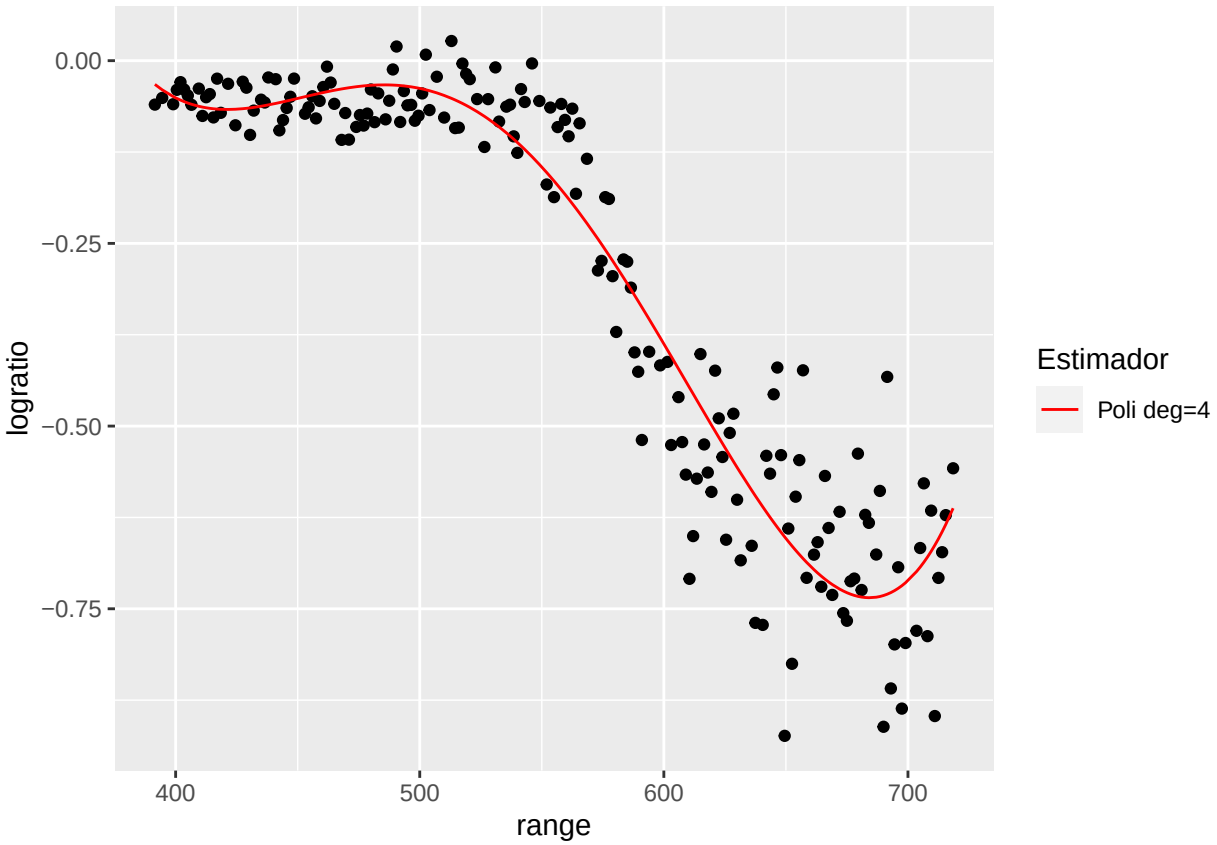
Separamos un 20% de los datos elegidos al azar.

```
indices=sample(length(datos$range),size=0.2*length(datos$range))
test_set=datos[indices,]
training_set=datos[-indices,]
```

1. Por el método de mínimos cuadrados realice un ajuste de un modelo lineal polinómico de grado 4 en **range**. Grafique los puntos observados y la función de regresión estimada.

Resolución:

```
poli4=lm(training_set$logratio ~ poly(training_set$range, 4, raw=TRUE))
# otra forma
# poli4=predict(lm(formula=training_set$logratio ~ range+range2+range3+range4))
# esto devuelve el vector de logratios estimados
# poli4=function(x){
#   betas=lm(formula=training_set$logratio ~ range+range2+range3+range4)$coefficients
#   sum(betas*c(1,x,x^2,x^3,x^4))
# }
datos_plot=data.frame(cbind('Range'=training_set$range,
                            'Lineal_4'=predict(poli4)))
datos_plot %>%
  ggplot()+
  geom_point(data=training_set,aes(x=range,y=logratio))+
  geom_line(aes(x=Range,y=Lineal_4,col='Poli deg=4'))+
  scale_color_manual("Estimador",
                     breaks=c('Poli deg=4'),
                     values=c('red'))
```



HACERLO TAMBIEN CON LA FORMULA DE BETA! VER LA FOTO DE JEMI!

2. Repita el item anterior considerando un modelo lineal polinómico de grado 9 y uno de grado 10 en **range**. Grafique y compare.

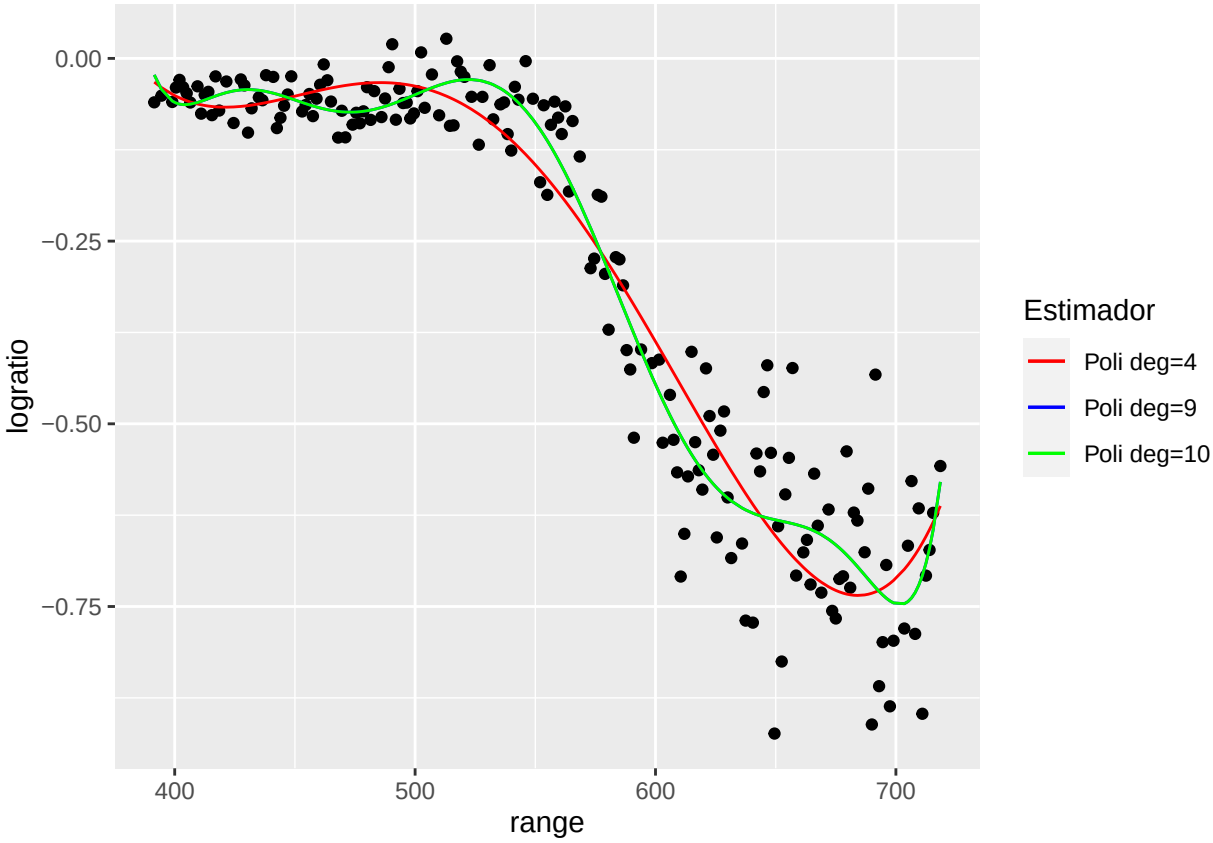
Resolución:

```
range=training_set$range
range2=range^2
range3=range^3
range4=range^4
range5=range^5
range6=range^6
range7=range^7
range8=range^8
range9=range^9
range10=range^10
# poli9=function(x){
# betas=lm(formula=training_set$logratio ~ range+range2+range3+range4+
# range5+range6+range7+range8+range9)$coefficients
##################################################
# Una forma de escribir lm. Uso esta
poli9=lm(formula=training_set$logratio ~
           range+range2+range3+range4+range5+range6+range7+range8+range9)
# Otra.
poli9_2=lm(training_set$logratio ~ poly(training_set$range, 9, raw=TRUE))
names(poli9_2$coefficients)=c('(Intercept)',paste('range',1:9,sep=""))
# Otra
```

```r
poli10=lm(training_set$logratio ~ poly(training_set$range, 10, raw=TRUE))

datos_plot=data.frame(cbind('Range'=training_set$range,
                            'Lineal_4'=predict(poli4),
                            'Lineal_9'=predict(poli9),
                            'Lineal_10'=predict(poli10)))

datos_plot %>%
  ggplot()+
  geom_point(data=training_set,aes(x=range,y=logratio))+
  geom_line(aes(x=Range,y=Lineal_4,col='Poli deg=4'))+
  geom_line(aes(x=Range,y=Lineal_9,col='Poli deg=9'))+
  geom_line(aes(x=Range,y=Lineal_10,col='Poli deg=10'))+
  scale_color_manual("Estimador",
                     breaks=c('Poli deg=4','Poli deg=9','Poli deg=10'),
                     values=c('red','blue','green'))
```



3. Considerando polinómios de grado 1 a 10 y elija con cúal propone predecir.

Resolución:

Calculemos el error cuadrático medio con nuestro testing set para ver que metodo polinomial aproxima mejor. Notar que para usar la función predict() el nuevo set de datos test_set tiene que tener **tantas variables como nuetro predictor y con el mismo nombre**.

```r
# Una forma
test_set=datos[indices,]
test_set$range1=test_set$range
```

```
test_set$range2=test_set$range^2
test_set$range3=test_set$range^3
test_set$range4=test_set$range^4
test_set$range5=test_set$range^5
test_set$range6=test_set$range^6
test_set$range7=test_set$range^7
test_set$range8=test_set$range^8
test_set$range9=test_set$range^9
test_set$range10=test_set$range^10
# Errores.
mean((test_set$logratio-predict(poli9,newdata=test_set))^2)
```

```
## Warning in predict.lm(poli9, newdata = test_set): prediction from a rank-
## deficient fit may be misleading
```

```
## [1] 0.01064073
```

```
mean((test_set$logratio-predict(poli10,newdata=test_set))^2)
```

```
## Warning: 'newdata' had 44 rows but variables found have 177 rows
```

```
## Warning in predict.lm(poli10, newdata = test_set): prediction from a rank-
## deficient fit may be misleading
```

```
## Warning in test_set$logratio - predict(poli10, newdata = test_set): longitud de
## objeto mayor no es múltiplo de la longitud de uno menor
```

```
## [1] 0.1445249
```

```
#####################################
# Otra forma
test_set_2=datos[indices,]
# Arreglamos test_set para que coincida con las variables del predictor
for(i in 1:9){
  # Arranco en i=2 porque range^1 ya está en test_set
  test_set_2[,i+2]=test_set_2$range^i
}
names(test_set_2)=c('range','logratio',paste('range',1:9,sep=""))
# Errores.
mean((test_set_2$logratio-predict(poli9,newdata=test_set_2))^2) # este anda
```

```
## Warning in predict.lm(poli9, newdata = test_set_2): prediction from a rank-
## deficient fit may be misleading
```

```
## [1] 0.01064073
```

```
mean((test_set_2$logratio-predict(poli9_2,newdata=test_set_2))^2) # este no anda
```

```
## Warning: 'newdata' had 44 rows but variables found have 177 rows
```

```
## Warning in predict.lm(poli9_2, newdata = test_set_2): prediction from a rank-
## deficient fit may be misleading
```

```
## Warning in test_set_2$logratio - predict(poli9_2, newdata = test_set_2):
## longitud de objeto mayor no es múltiplo de la longitud de uno menor
```

```
## [1] 0.1445249
```

```
mean((test_set_2$logratio-predict(poli10,newdata=test_set_2))^2) # este no anda
```

```
## Warning: 'newdata' had 44 rows but variables found have 177 rows
```

```
## Warning in predict.lm(poli10, newdata = test_set_2): prediction from a rank-
## deficient fit may be misleading
```

```
## Warning in test_set_2$logratio - predict(poli10, newdata = test_set_2): longitud
## de objeto mayor no es múltiplo de la longitud de uno menor
```

```
## [1] 0.1445249
```

EL METODO CON EL COMANDO POLY() ME DA ERROR PORQUE TIENEN DISTINA LONGITUD, QUE HAGO? ADEMAS ME DA UN ERROR GIGANTE COMPARADO AL QUE LE DIO A JEMI. ¿? TAMPOCO ANDA PONIENDOLE EL MISMO NOMBRE A LOS COEF DE POLI9 Y A TEST_SET!

PENSAR COMO ITERARLO! ARMAR UNA MATRIZ Y USAR LM SOBRE ESA MATRIZ SACANDOLE COLUMNAS. O USAR

4. Considere polinomios hasta grado 10, regresión por núcleos Gaussianos y vecinos más cercanos. Determine en cada caso como predecir. Luego, evalúe los procedimientos propuestos en el conjunto de testeo para elegír el método que utilizaría en un nuevo valor de x.

Resolución: