Statistical learning and deep learning: theoretical background and hands-on sessions

Lezione 2

S. Biffani

IBBA/CNR

27 gennaio, 2023

Regressione Lineare - Quick Recap 1

1. metodo semplice e potente per predire una risposta lineare (Y) a partire da una o più variabili indipendenti ($X_1, ..., X_n$)
2. può sembrare *vetusto* ma è tuttora estremamente efficace e facile da interpretare
3. tra i migliori strumenti per fare sia **inferenza** che **predizione**
   - c'é un rapporto tra la mia Y e le diverse X?
   - quanto è forte?
   - Quale X contribuisce di più?
   - Posso predire la Y? con che accuratezza?

Regressione Lineare - Quick Recap 2

- algoritmo *supervised*
  - impara a modellare la risposta (Y) in funzione di alcune *features* (X_s) identificando una linea (od una superficie) che *aggiusta* meglio i dati
- predire il prezzo di una casa sulla base del numero di stanze (. . . e del quartiere)
- predire la produzione di latte sulla base dell'ordine di parto

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_p x_p + \epsilon$$

β_i = le fondamenta del nostro modello. Rappresenta ciò che il nostro modello *impara* (*learn*) attraverso i dati ed attraverso la funzione utilizzata (lineare in questo caso)

Regressione Lineare - Quick Recap 3

Una volta *stimati* i parametri, posso fare una previsione

$$\hat{y} = \hat{\beta_0} + \hat{\beta_1 x_1} + \hat{\beta_2 x_2} + ... + \hat{\beta_p x_p} + \epsilon$$

Esempio: Stimare il prezzo di una casa sulla base della sua dimensione in mq $prezzo = \beta 0 + \beta_1 * mq$

quale può essere la stima più semplice ?

Regressione Lineare - Quick Recap 4

Il prezzo medio $= prezzo = 290000 + 0 * mq$

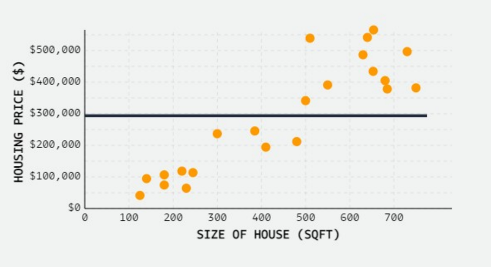


Figura 1: esempio regressione: prezzo di una casa in funzione della dimensione

non mi sembra un buona soluzione, soprattutto se guardo agli errori singoli

Regressione Lineare - Quick Recap 4

$\epsilon_i = f(x_i) - \hat{f}_i(x) = y_i - \hat{y}_i$

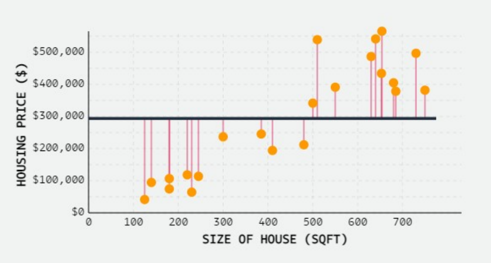


Figura 2: esempio regressione: prezzo di una casa in funzione della dimensione

Somma dei Residui: $RSS = \sum_{i=1}^{n} \epsilon_i^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$

Regressione Lineare - Quick Recap 5

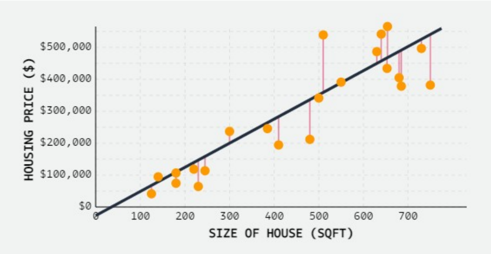


Figura 3: esempio regressione: prezzo di una casa in funzione della dimensione

Devo trovare i coefficienti β_0 e β_1 che minimizzano:

Somma dei Residui: $RSS = \sum_{i=1}^{n} \epsilon_i^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$

Regressione Lineare - Quick Recap 6

Come valutare la bontà del mio modello ? **loss function**

- funzione che calcola la distanza tra Y e $\hat{Y}$

$$MSE = 1/n \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \text{L2 regularization}$$

o la sua radice

$$RMSE = \sqrt{1/n \sum_{i=1}^{n} (y_i - \hat{y}_i)^2}$$

Regressione Lineare - Quick Recap 7

Come valutare la bontà del mio modello ? **R-Quadro**

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \overline{y})^2}$$

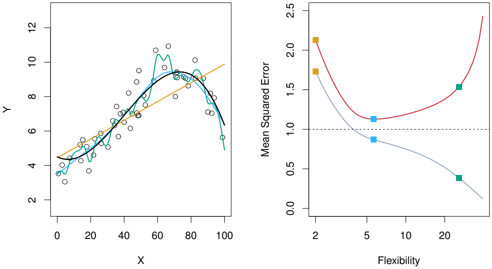
Che proporzione della varianza totale $(\sum_{i=1}^{n}(y_i - \overline{y})^2)$ è spiegata dal nostro modello

Regressione Lineare - Quick Recap 8

domanda: scegliereste il vostro modello sulla base del solo R^2?

Regressione Lineare - Quick Recap 9

ATTENZIONE



- Siamo nei *training data*: l'errore diminuisce all'aumentare del numero di predittori
- Esistono parametri alternativi: C_p, *Akaike Information Criterion* (AIC), *Bayesian Information Criterion* (BIC) e adjusted R^2.
- Penalizzano tutti l'RSS originale in funzione del numero di predittori

Regressione Lineare - Quick Recap 10

Quando abbiamo una regressione lineare *semplice* β_0 e β_1, possono essere ottenuti con metodi diretti e corrispondono a :

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sum_{i=1}^{n}(x_i - \overline{x})^2}$$

$$\hat{\beta}_0 = \overline{y} - \hat{\beta}_1 \overline{x}$$

Nel caso di regressioni lineari a più parametri dobbiamo ricorrere a metodi cosidetti *iterativi* (e.g. *gradient descent*)

Regressione Lineare - Quick Recap 11

- Esempio

Librerie utilizzate: *MASS* (Nativa), *ISLR2* (da installare) e *tidyverse* (da installare)

```
#install.packages('ISLR2')
library(MASS)
library(ISLR2)
library(tidyverse)
```

Useremo il **Boston** data set che contiene informazioni sul valore delle case (**medv**) e cercheremo di predirlo usando 12 predittori

Regressione Lineare - Quick Recap 12

Esempio di regressione lineare

```r
lm.fit <- lm(medv~lstat , data = Boston)
lm.fit
```

```
##
## Call:
## lm(formula = medv ~ lstat, data = Boston)
##
## Coefficients:
## (Intercept)        lstat
##       34.55        -0.95
```

Regressione Lineare - Quick Recap 13

Risultato

```
summary(lm.fit)
```

```
##
## Call:
## lm(formula = medv ~ lstat, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.168  -3.990  -1.318   2.034  24.500
##
## Coefficients:
##             Estimate Std. Error t value            Pr(>|t|)
## (Intercept) 34.55384    0.56263   61.41 <0.0000000000000002 ***
## lstat       -0.95005    0.03873  -24.53 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.216 on 504 degrees of freedom
## Multiple R-squared:  0.5441, Adjusted R-squared:  0.5432
## F-statistic: 601.6 on 1 and 504 DF,  p-value: < 0.00000000000000022
```

Regressione Lineare - Quick Recap 13

Rapporto tra X e Y

```
ggplot(Boston,aes(lstat,medv))+
  geom_point()+
  geom_smooth(method = 'lm')
```

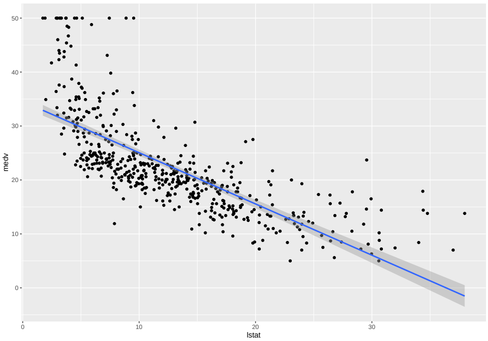


Figura 4: X = lstat, Y = medv

Si osserva un rapporto non lineare

Regressione Lineare - Quick Recap 14

Plot dei residui

```
data.frame(R=residuals(lm.fit),
           P=predict(lm.fit))-> X
ggplot(X,aes(P,R))+
  geom_point()-> lineare
lineare
```

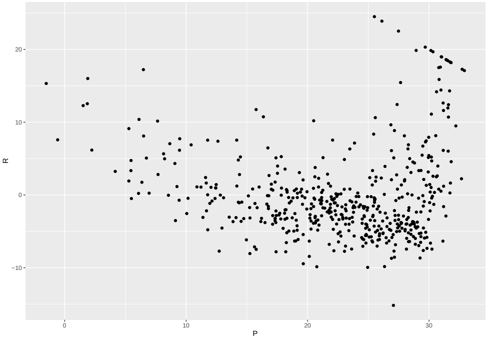


Figura 5: X = Valori Predetti dal Modello, Y = Residui del modello

Regressione Lineare - Quick Recap 15

Trasformazione non lineare dei predittori

```r
lm.fit2 <- lm(medv ~ lstat + I(lstat^2), data=Boston)
summary(lm.fit2)
```

```
##
## Call:
## lm(formula = medv ~ lstat + I(lstat^2), data = Boston)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.2834  -3.8313  -0.5295   2.3095  25.4148
##
## Coefficients:
##              Estimate Std. Error t value            Pr(>|t|)
## (Intercept) 42.862007   0.872084   49.15 <0.0000000000000002 ***
## lstat       -2.332821   0.123803  -18.84 <0.0000000000000002 ***
## I(lstat^2)   0.043547   0.003745   11.63 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.524 on 503 degrees of freedom
## Multiple R-squared:  0.6407, Adjusted R-squared:  0.6393
## F-statistic: 448.5 on 2 and 503 DF,  p-value: < 0.00000000000000022
```

Regressione Lineare - Quick Recap 16

Trasformazione non lineare dei predittori - uso funzione *poly*

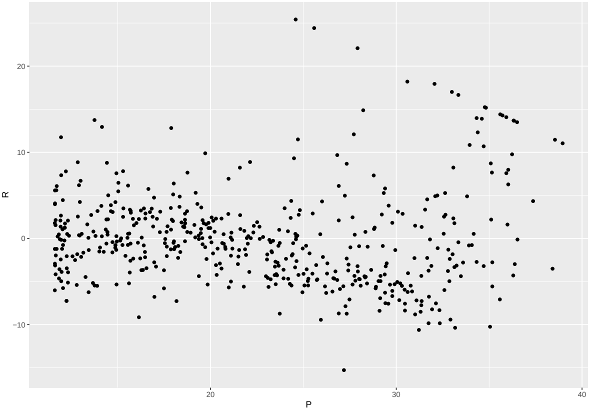
```r
lm.fit3 <- lm(medv ~ poly(lstat,2,raw=T), data=Boston)
summary(lm.fit3)
```

```
##
## Call:
## lm(formula = medv ~ poly(lstat, 2, raw = T), data = Boston)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.2834  -3.8313  -0.5295   2.3095  25.4148
##
## Coefficients:
##                           Estimate Std. Error t value             Pr(>|t|)
## (Intercept)              42.862007   0.872084   49.15 <0.0000000000000002 ***
## poly(lstat, 2, raw = T)1 -2.332821   0.123803  -18.84 <0.0000000000000002 ***
## poly(lstat, 2, raw = T)2  0.043547   0.003745   11.63 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.524 on 503 degrees of freedom
## Multiple R-squared:  0.6407, Adjusted R-squared:  0.6393
## F-statistic: 448.5 on 2 and 503 DF,  p-value: < 0.00000000000000022
```

Regressione Lineare - Quick Recap 17

Ricontrolliamo ora il plot dei residui

```
data.frame(R=residuals(lm.fit3),
           P=predict(lm.fit3))-> X
ggplot(X,aes(P,R))+
  geom_point()-> quad
quad
```



Regressione Lineare - Quick Recap 18

un approccio *tidy*: `library(broom)`

funzione: `augment`

```
library(broom)
augment(lm.fit) %>%
  head()
```

```
## # A tibble: 6 x 8
##    medv lstat .fitted .resid    .hat .sigma   .cooksd .std.resid
##   <dbl> <dbl>   <dbl>  <dbl>   <dbl>  <dbl>     <dbl>      <dbl>
## 1  24    4.98    29.8 -5.82  0.00426   6.22 0.00189       -0.939
## 2  21.6  9.14    25.9 -4.27  0.00246   6.22 0.000582      -0.688
## 3  34.7  4.03    30.7  3.97  0.00486   6.22 0.00100        0.641
## 4  33.4  2.94    31.8  1.64  0.00564   6.22 0.000198       0.264
## 5  36.2  5.33    29.5  6.71  0.00406   6.21 0.00238        1.08
## 6  28.7  5.21    29.6 -0.904 0.00413   6.22 0.0000440     -0.146
```

l'oggetto creato (tibble) contiene molte delle informazioni di cui abbiamo bisogno

Funzioni e Iterazioni 1

funzione

‘nomeFunzione <-
function(lista_argomenti){
comando1
return(valore) }‘

```
my_fun2 <- function( a , b , c) {
    y <- a**b
return(y + a*b + c)
}

my_fun2(10,5,-2)
```

```
## [1] 100048
```

Funzioni e Iterazioni 2

Loop

```
for (i in sequenza) {
comando }
```

```
for (x in 1:5) {
    print(x)
}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
```

```
vect_1 <- c(2, 7, 4, 9, 8)
vect_2 <- numeric()

for(num in vect_1) {
    vect_2 <- c(vect_2, num * 10)
}

vect_2
```

```
## [1] 20 70 40 90 80
```