Lecture 8: Machine Learning
Mas allá de la linealidad

Big Data and Machine Learning en el Mercado Inmobiliario
Educación Continua

Ignacio Sarmiento-Barbieri

Universidad de los Andes

March 31, 2022

Agenda

1. Recap
   - Predicción y Overfit
2. Selección de Modelos y Regularización
3. Selección de Modelos y Regularización
   - Lasso
   - Ridge
4. Elastic net
5. Break

Overfit y Predicción fuera de Muestra

- ML nos interesa la predicción fuera de muestra
- Overfit: modelos complejos predicen muy bien dentro de muestra, pero tienden a hacer un mal trabajo fuera de muestra
- Hay que elegir el modelo que "mejor" prediga
  - Métodos de Remuestreo
    - Enfoque del conjunto de validación
    - Loocv
    - Validación cruzada en K-partes (5 o 10)

Selección de Modelos: Motivación

$$Precio = \beta_0 + \beta_1 Habitaciones + \epsilon \tag{1}$$

```
mean_squared_error(y_test,y_pred):2.648756376377227e+18
```

$$Precio = \beta_0 + \beta_1 Habitaciones + \beta_2 Superficie + \epsilon \tag{2}$$

```
mean_squared_error(y_test,y_pred2): 2.520184992914835e+18
```

$$Precio = \beta_0 + \beta_1 Habitaciones + \beta_2 Superficie + \beta_3 Dormitorios + \epsilon \tag{3}$$

```
mean_squared_error(y_test,y_pred3): 2.232920478011764e+18
```

Selección de Modelos: Motivación

- Tenemos M_k modelos
- Queremos encontrar el que mejor predice fuera de muestra
- Hay distintas formas de enfrentarlo
- Las clásicas
  - Elección del mejor conjunto
  - Elección por pasos
    - Hacia adelante (Forward selection)
    - Hacia atras (Backward selection)

Regularización

Lasso

- Para un $\lambda \geq 0$ dado, consideremos el siguiente problema de optimización

$$min_{\beta} E(\beta) = \sum_{i=1}^{n} (y_i - \beta_0 - x_{i1}\beta_1 - \cdots - x_{ip}\beta_p)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \tag{4}$$

Lasso

- Para un $\lambda \geq 0$ dado, consideremos el siguiente problema de optimización

$$min_{\beta} E(\beta) = \sum_{i=1}^{n} (y_i - \beta_0 - x_{i1}\beta_1 - \cdots - x_{ip}\beta_p)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \tag{4}$$

  - "LASSO's free lunch": selecciona automáticamente los predictores que van en el modelo ($\beta_j \neq 0$) y los que no ($\beta_j = 0$)
  - Porque? Los coeficientes que no van son soluciones de esquina
  - $L(\beta)$ es no differentiable

Lasso Intuición en 1 Dimension

- Lasso Intuición

$$min_{\beta} E(\beta) = \sum_{i=1}^{n} (y_i - x_i\beta)^2 + \lambda|\beta| \tag{5}$$

- Un solo predictor, un solo coeficiente
- Si $\lambda = 0$

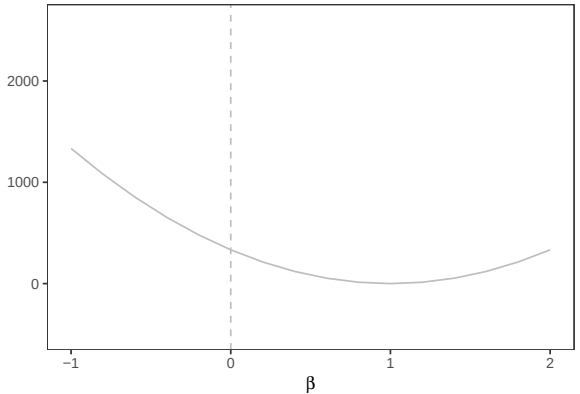
$$min_{\beta} E(\beta) = \sum_{i=1}^{n} (y_i - x_i\beta)^2 \tag{6}$$

- y la solución es

$$\hat{\beta}_{OLS} \tag{7}$$

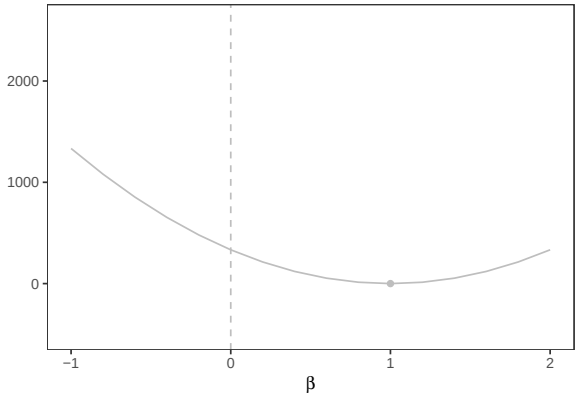
Intuición en 1 Dimension

$$min_{\beta} E(\beta) = \sum_{i=1}^{n} (y_i - x_i\beta)^2 + \lambda|\beta| \tag{8}$$



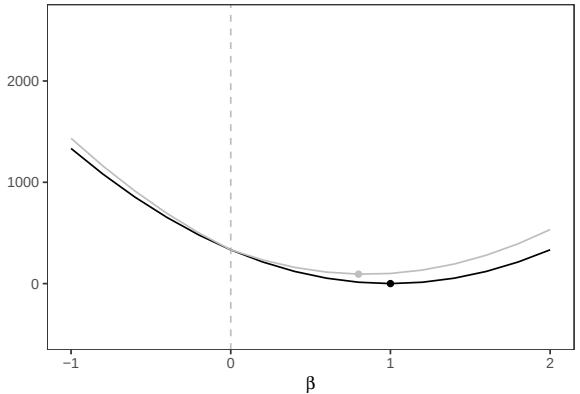
Intuición en 1 Dimension

$$min_{\beta} E(\beta) = \sum_{i=1}^{n} (y_i - x_i\beta)^2 + \lambda|\beta| \tag{9}$$



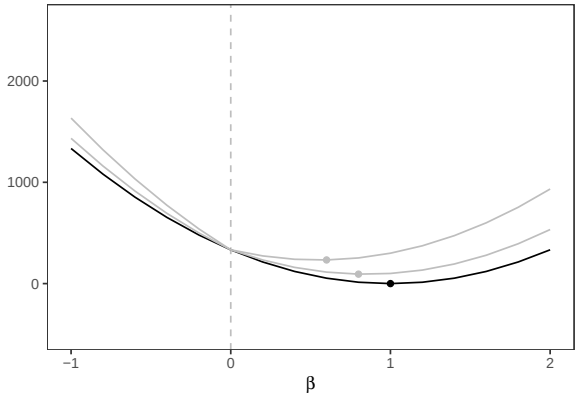
Intuición en 1 Dimension

$$min_{\beta} E(\beta) = \sum_{i=1}^{n} (y_i - x_i\beta)^2 + \lambda|\beta| \tag{10}$$



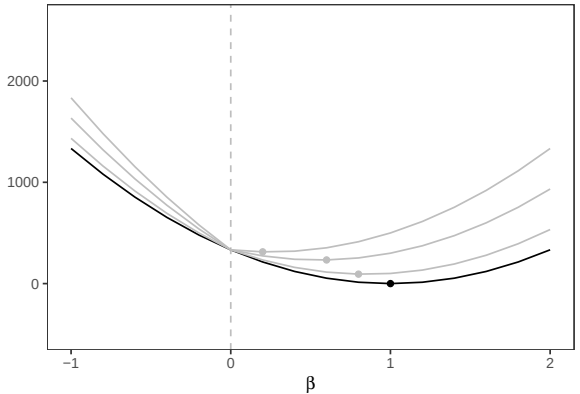
Intuición en 1 Dimension

$$min_{\beta} E(\beta) = \sum_{i=1}^{n} (y_i - x_i\beta)^2 + \lambda|\beta| \tag{11}$$



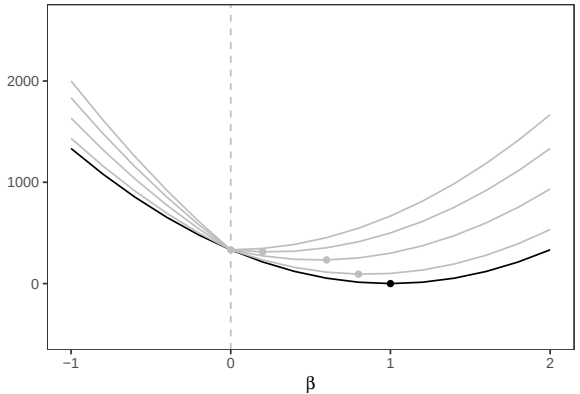
Intuición en 1 Dimension

$$min_\beta E(\beta) = \sum_{i=1}^{n} (y_i - x_i\beta)^2 + \lambda|\beta| \tag{12}$$



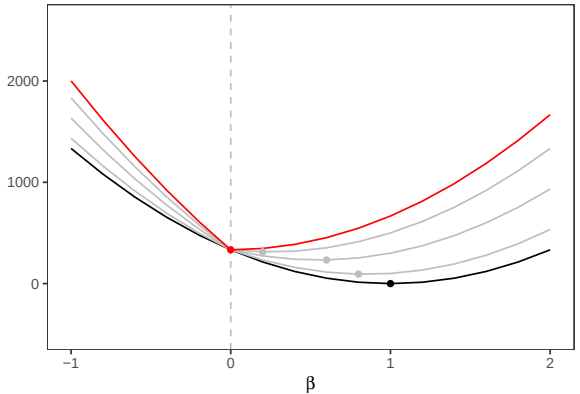
Intuición en 1 Dimension

$$min_\beta E(\beta) = \sum_{i=1}^{n} (y_i - x_i\beta)^2 + \lambda|\beta| \tag{13}$$



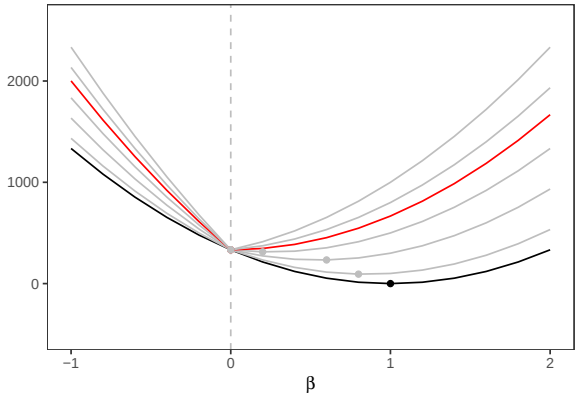
Intuición en 1 Dimension

$$min_\beta E(\beta) = \sum_{i=1}^{n} (y_i - x_i\beta)^2 + \lambda|\beta| \tag{14}$$



Intuición en 1 Dimension

$$min_\beta E(\beta) = \sum_{i=1}^{n} (y_i - x_i\beta)^2 + \lambda|\beta| \tag{15}$$



Intuición en 1 Dimension

Ejemplo en Python

```
from sklearn.linear_model import Lasso
```

```
# define model
model1 = Lasso(alpha=0.01).fit(X_train[['rooms']],y_train)
print(np.round_(model1.coef_,decimals=2))
```

```
[2093765.47]
```

Intuición en 1 Dimension

Ejemplo en Python

```
# define model
model2 = Lasso(alpha=1000000).fit(X_train[['rooms']],y_train)
print(np.round_(model2.coef_,decimals=2))
```

```
[1983417.37]
```

```
# define model
model3 = Lasso(alpha=10000000).fit(X_train[['rooms']],y_train)
print(np.round_(model3.coef_,decimals=2))
```

```
[990284.48]
```

```
model4 = Lasso(alpha=100000000000000000).fit(X_train[['rooms']],y_train)
print(np.round_(model4.coef_,decimals=2))
```

```
[0.]
```

Intuición en 1 Dimension

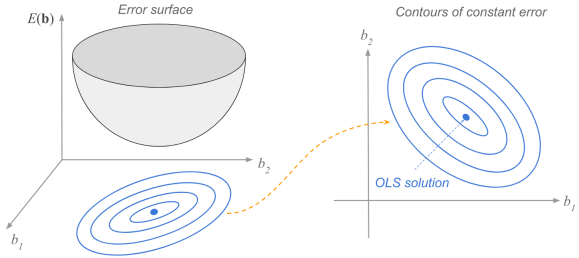
$$min_{\beta} E(\beta) = \sum_{i=1}^{n} (y_i - x_i\beta)^2 + \lambda|\beta| \tag{16}$$

la solución analítica es

$$\hat{\beta}_{lasso} = \begin{cases} 0 & \text{si } \lambda \geq \lambda^* \\ \hat{\beta}_{OLS} - \frac{\lambda}{2} & \text{si } \lambda < \lambda^* \end{cases} \tag{17}$$

Intuición en 2 Dimensiones (OLS)

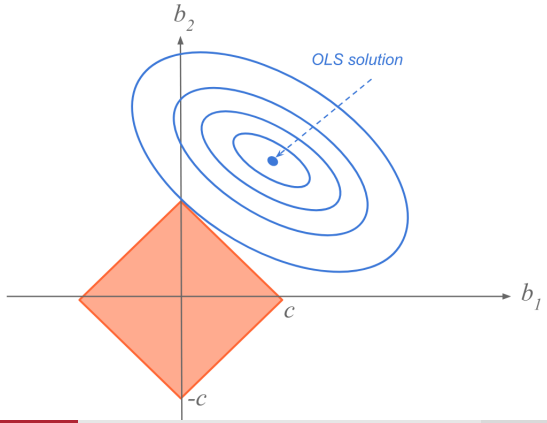
$$min_{\beta} E(\beta) = \sum_{i=1}^{n} (y_i - x_{i1}\beta_1 - x_{i1}\beta_2)^2 \tag{18}$$



Fuente: https://allmodelsarewrong.github.io

Intuición en 2 Dimensiones (Lasso)

$$min_{\beta} E(\beta) = \sum_{i=1}^{n} (y_i - x_{i1}\beta_1 - x_{i1}\beta_2)^2 \text{ s.a } (|\beta_1| + |\beta_2|) \leq c \tag{19}$$



Ridge

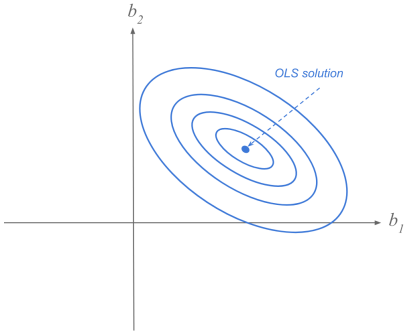
- Para un $\lambda \geq 0$ dado, consideremos ahora el siguiente problema de optimización

$$min_{\beta} E(\beta) = \sum_{i=1}^{n} (y_i - \beta_0 - x_{i1}\beta_1 - \cdots - x_{ip}\beta_p)^2 + \lambda \sum_{j=1}^{p} (\beta_j)^2 \tag{20}$$

- La intuición es similar a lasso, pero la vamos a extender a 2-Dim

Intuición en 2 Dimensiones (OLS)

$$min_{\beta} E(\beta) = \sum_{i=1}^{n} (y_i - x_{i1}\beta_1 - x_{i1}\beta_2)^2 \tag{21}$$



Fuente: https://allmodelsarewrong.github.io

Intuición en 2 Dimensiones (Ridge)

- Al problema

$$min_{\beta} E(\beta) = \sum_{i=1}^{n} (y_i - \beta_0 - x_{i1}\beta_1 - \cdots - x_{ip}\beta_p)^2 + \lambda \sum_{j=1}^{p} (\beta_j)^2 \tag{22}$$

- podemos escribirlo como

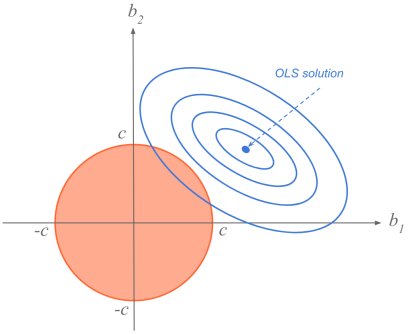
$$min_{\beta} E(\beta) = \sum_{i=1}^{n} (y_i - x_{i1}\beta_1 - x_{i1}\beta_2)^2 \tag{23}$$

sujeto a

$$\left((\beta_1)^2 + (\beta_2)^2\right) \leq c$$

Intuición en 2 Dimensiones (Ridge)

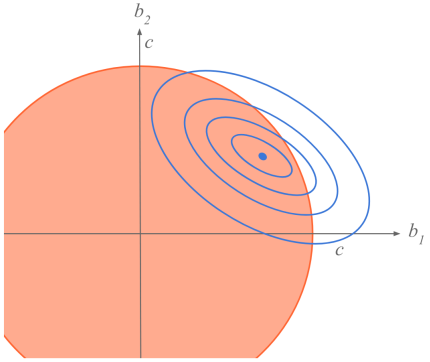
$$min_{\beta} E(\beta) = \sum_{i=1}^{n} (y_i - x_{i1}\beta_1 - x_{i1}\beta_2)^2 \text{ s.a } \left((\beta_1)^2 + (\beta_2)^2\right) \leq c \tag{24}$$



Fuente: https://allmodelsarewrong.github.io

Intuición en 2 Dimensiones (Ridge)

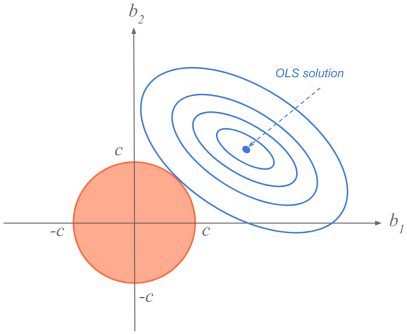
$$min_{\beta} E(\beta) = \sum_{i=1}^{n} (y_i - x_{i1}\beta_1 - x_{i1}\beta_2)^2 \text{ s.a } \left((\beta_1)^2 + (\beta_2)^2\right) \leq c \tag{25}$$



Fuente: https://allmodelsarewrong.github.io

Intuición en 2 Dimensiones (Ridge)

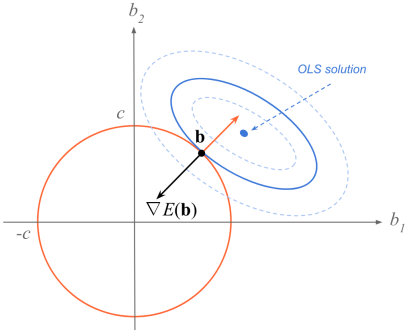
$$min_{\beta} E(\beta) = \sum_{i=1}^{n} (y_i - x_{i1}\beta_1 - x_{i1}\beta_2)^2 \text{ s.a } \left((\beta_1)^2 + (\beta_2)^2\right) \leq c \tag{26}$$



Fuente: https://allmodelsarewrong.github.io

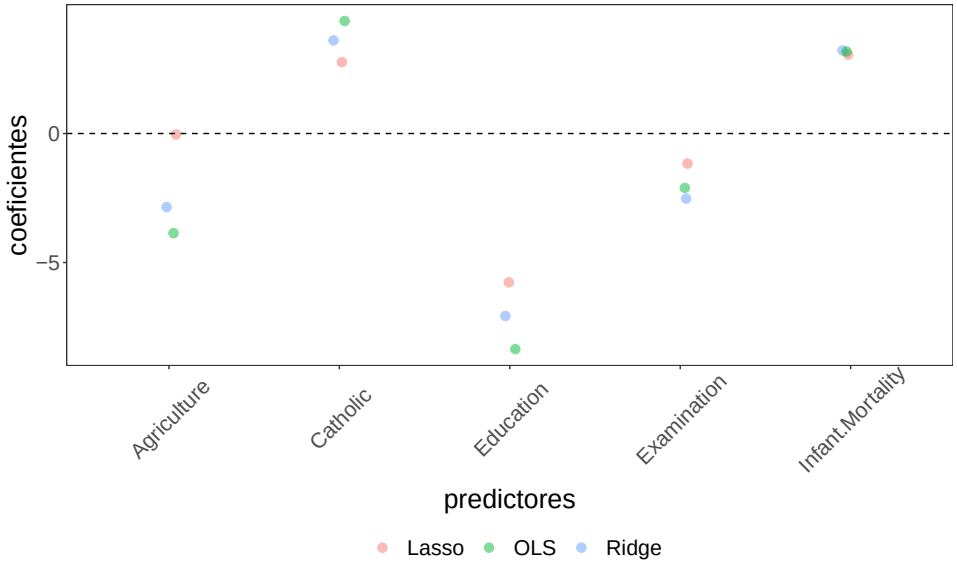
Intuición en 2 Dimensiones (Ridge)

$$min_{\beta} E(\beta) = \sum_{i=1}^{n} (y_i - x_{i1}\beta_1 - x_{i1}\beta_2)^2 \text{ s.a } \left((\beta_1)^2 + (\beta_2)^2\right) \leq c \tag{27}$$



Fuente: https://allmodelsarewrong.github.io

Lasso y Ridge Ejemplo



Comentarios técnicos

- Lasso y ridge son sesgados, pero las disminuciones en varianza pueden compensar estoy y llevar a un MSE menor
- Lasso encoje a cero, Ridge no tanto
- Importante para aplicación:
  - Estandarizar los datos (media 0, y varianza 1)
  - Como elegimos λ?

Comentarios técnicos: selección de λ

- Como elegimos λ?
- λ es un parámetro y lo elegimos usando validación cruzada
  1. Partimos la muestra de entrenamiento en K Partes: $M_{train} = M_{fold\,1} \cup M_{fold\,2} \cdots \cup M_{fold\,K}$
  2. Cada conjunto $M_{fold\,K}$ va a jugar el rol de una muestra de evaluación $M_{eval\,k}$. Entonces para cada muestra
     - $M_{train-1} = M_{train} - M_{fold\,1}$
     - $\vdots$
     - $M_{train-k} = M_{train} - M_{fold\,k}$
  3. Luego hacemos el siguiente loop
     1. Para $\lambda_i = 0, 0.001, 0.002, \ldots, \lambda_{max}$
     - Para $k = 1, \ldots, K$
       - Ajustar el modelo $m_{i,k}$ con λ_i en $M_{train-k}$
       - Calcular y guardar el $MSE(m_{i,k})$ usando M_{eval-k}
     - fin para k
     - Calcular y guardar $MSE_i = \frac{1}{K} MSE(m_{i,k})$
     2. fin para λ
  4. Encontrar el menor MSE_i y usar ese $\lambda_i = \lambda^*$

Elastic net

$$min_{\beta} NEL(\beta) = \sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \alpha \left(\lambda_1 \sum_{j=1}^{p} |\beta_j| \right) + (1-\alpha) \left(\lambda_2 \sum_{j=1}^{p} (\beta_j)^2 \right) \quad (28)$$

- Si $\alpha = 1$ Lasso
- Si $\alpha = 0$ Rigdge
- $\hat{\beta}_{EN} = \frac{1}{\sqrt{1+\lambda_2}} \hat{\beta}_{naive\, EN}$

Volvemos en 5 mins con Python