Machine Learning Supervisado: Regularización y Tuning de Hiperparámetros

HE2: Consultoría Económica con IA Responsable

Santiago Neira & Catalina Bernal

Universidad de los Andes
Departamento de Economía

5 de febrero de 2026

Agenda de hoy

1. Recorderis: Lo Esencial de la Clase Anterior
2. Ejercicios de Intuición
3. El Problema: Overfitting en OLS
4. Hiperparámetros: ¿Qué Son y Por Qué Importan?
5. Ridge Regression (Regularización L2)
6. Lasso Regression (Regularización L1)
7. Elastic Net: Lo Mejor de Ambos Mundos
8. Tuning de Hiperparámetros
9. Manos a la Obra

Recorderis: El Cambio de Paradigma

Econometría Clásica

- Interés en $\hat{\beta}_j$
- ¿X causa Y?
- Queremos $\hat{\beta}$ **insesgados**
- Supuestos fuertes

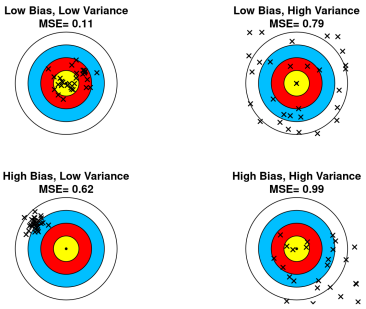
Machine Learning

- Interés en $\hat{Y}$
- ¿Qué tan bien predecimos?
- Queremos **minimizar error total**
- A veces conviene introducir sesgo

La idea clave de hoy

Regularización = introducir sesgo deliberadamente para reducir varianza y mejorar predicción fuera de muestra.

Recorderis: Bias-Variance Tradeoff



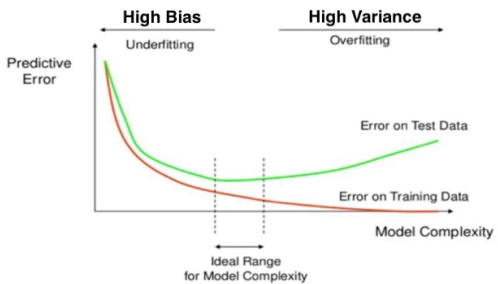
Descomposición del error:

$$\text{Error} = \text{Bias}^2 + \text{Varianza} + \text{Irreducible}$$

- **Alto Bias:** Modelo muy simple, no captura la señal
- **Alta Varianza:** Modelo muy complejo, memoriza el ruido

No podemos minimizar ambos simultáneamente.

Recorderis: La U-Curve



Observaciones clave:

- Error de training **siempre** decrece con complejidad
- Error de test tiene forma de **U**
- El óptimo está en el **mínimo** de la curva de test
- La brecha indica **overfitting**

Recorderis: Train/Test y Cross-Validation

¿Por qué dividir los datos?

- El error de entrenamiento es **optimista**
- Necesitamos evaluar en datos **nunca vistos**
- El test set simula las condiciones de producción

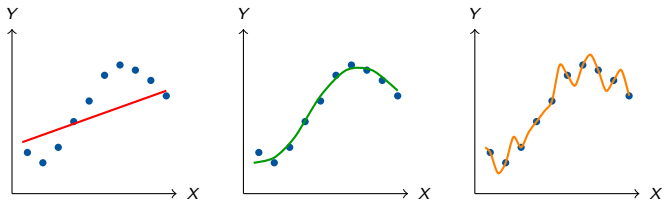
Cross-Validation:

$$CV_{(K)} = \frac{1}{K} \sum_{k=1}^{K} MSE_k$$

Pregunta de hoy: ¿Cómo usamos CV para elegir la **cantidad óptima de regularización**?

Ejercicio 1: ¿Cuál modelo elegirías?

Contexto: Tres analistas ajustaron modelos a los **mismos datos** de entrenamiento.

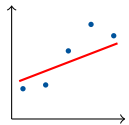


Pregunta para discutir

Si llegan **nuevos datos** del mismo fenómeno, ¿cuál modelo crees que predecirá mejor? ¿Por qué?

Ejercicio 1: Discusión

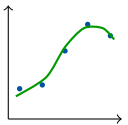
Modelo A



Underfitting

Alto sesgo, baja varianza

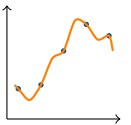
Modelo B



Balance

Captura señal, ignora ruido

Modelo C



Overfitting

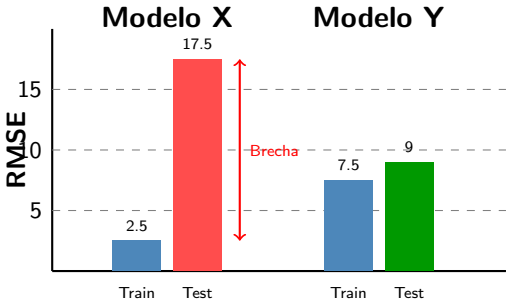
Bajo sesgo, alta varianza

Lección

El mejor modelo **no** es el que pasa más cerca de los puntos de entrenamiento. Es el que captura el **patrón subyacente** sin aprenderse el **ruido**.

Ejercicio 2: La Trampa del Error de Entrenamiento

Contexto: Evaluaste dos modelos. Aquí están sus errores:



Pregunta para discutir

El Modelo X tiene **menor error de entrenamiento**. ¿Es el mejor modelo? ¿Por qué?

Ejercicio 2: Discusión

Modelo X:

- RMSE training: **2.5**
- RMSE test: **17.5**
- Brecha: 15 ← ¡Peligro!

Memorizó los datos de entrenamiento pero no aprendió el patrón.

Modelo Y:

- RMSE training: **7.5**
- RMSE test: **9**
- Brecha: 1.5 ← Bien

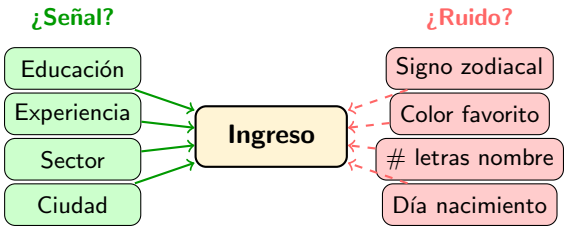
Captura el patrón y generaliza a datos nuevos.

Lecciones clave

1. El error de entrenamiento **siempre** es optimista
2. La **brecha** entre training y test indica overfitting
3. El mejor modelo tiene **menor error en datos nuevos**

Ejercicio 3: ¿Cuántas variables usar?

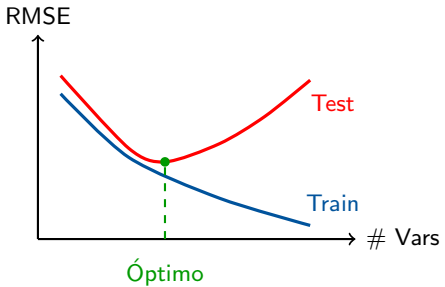
Contexto: Quieres predecir ingreso. Tienes 100 obs. y estas variables:



Pregunta para discutir

Si incluyes **todas** las variables, ¿qué pasa con el error de training? ¿Y con el error en datos nuevos?

Ejercicio 3: Discusión



Al añadir variables:

- Error training: siempre baja
- Error test: baja, luego sube

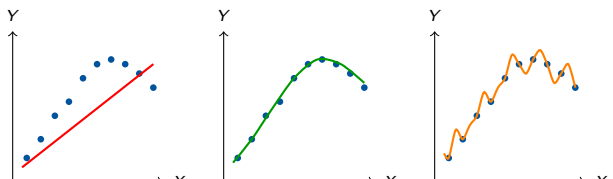
Las variables irrelevantes "ayudan" en training pero **añaden ruido** en datos nuevos.

Lección

Más variables $\neq$ mejor modelo. El óptimo balancea **señal** vs **ruido**.

Ejercicio 4: ¿Qué grado de polinomio usar?

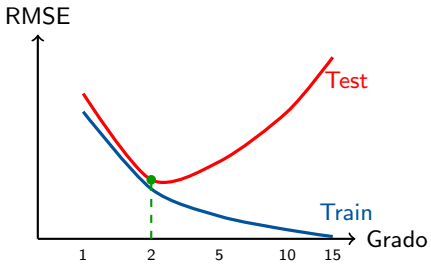
Contexto: Predices ventas (Y) con gasto en publicidad (X). Tienes 20 obs.



Pregunta para discutir

El polinomio de grado 15 pasa **más cerca** de todos los puntos. ¿Es el mejor para predecir?

Ejercicio 4: Discusión



Complejidad = # parámetros

- Grado 1: 2 parámetros
- Grado 2: 3 parámetros
- Grado 15: 16 parámetros

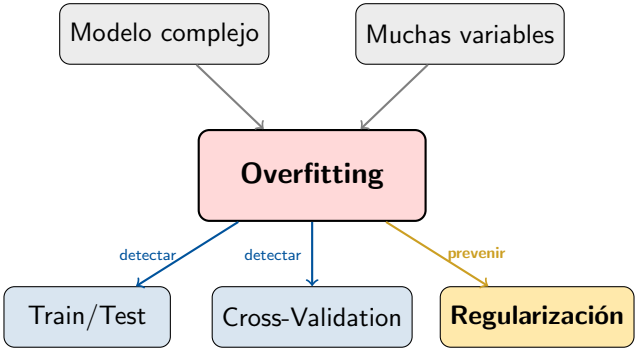
Con 20 obs. y 16 parámetros → demasiada flexibilidad.

Grado 2: captura la relación sin memorizar.

Lección

Más grado = más flexible = más riesgo de overfitting. Regularización permite modelos flexibles **penalizando** la complejidad.

Conectando con lo que viene...



Lo que viene hoy

Train/Test y CV **detectan** el problema. Regularización lo **previene** controlando la complejidad del modelo.

¿Qué pasa con OLS cuando hay muchas variables?

Recordemos OLS:

$$\hat{\boldsymbol{\beta}}_{OLS} = \arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} (Y_i - \mathbf{X}_i'\boldsymbol{\beta})^2 \tag{1}$$

Problemas cuando p (número de features) es grande:

- OLS ajusta *demasiado bien* los datos de training
- Los $\hat{\beta}_j$ pueden ser muy grandes (explotan)
- Alta varianza $\rightarrow$ pésima generalización
- Si $p > n$: ¡OLS ni siquiera tiene solución única!

La solución

Regularización: Añadir una penalización que *encoja* los coeficientes hacia cero.

La Idea de la Regularización

OLS minimiza solo el error:

$$\min_{\beta} \underbrace{\sum_{i=1}^{n} (Y_i - \mathbf{X}_i'\beta)^2}_{\text{Error de ajuste (RSS)}} \tag{2}$$

Regularización añade una penalización:

$$\min_{\beta} \underbrace{\sum_{i=1}^{n} (Y_i - \mathbf{X}_i'\beta)^2}_{\text{Error de ajuste}} + \underbrace{\lambda \cdot \text{Penalización}(\beta)}_{\text{Castigo por complejidad}} \tag{3}$$

- $\lambda \geq 0$ es el **hiperparámetro** de regularización
- $\lambda = 0 \rightarrow$ OLS (sin penalización)
- $\lambda \rightarrow \infty \rightarrow$ coeficientes $\rightarrow 0$

Parámetros vs. Hiperparámetros

Parámetros (β)

- Se **aprenden** de los datos
- El algoritmo los estima
- Ejemplo: $\beta_0, \beta_1, \ldots, \beta_p$

"¿Cuál es el efecto de X en Y?"

Hiperparámetros (λ, K, etc.)

- Se **eligen** antes de entrenar
- El usuario los especifica
- Ejemplo: λ en Ridge/Lasso, K en K-fold

"¿Cuánta regularización aplicamos?"

Pregunta clave

¿Cómo elegimos el valor óptimo de λ? $\rightarrow$ **Tuning de hiperparámetros**

Ridge Regression: Definición

Función objetivo de Ridge:

$$\hat{\boldsymbol{\beta}}^{Ridge} = \arg\min_{\boldsymbol{\beta}} \left\{ \sum_{i=1}^{n} (Y_i - \mathbf{X}_i'\boldsymbol{\beta})^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\} \tag{4}$$

En notación matricial:

$$\hat{\boldsymbol{\beta}}^{Ridge} = \arg\min_{\boldsymbol{\beta}} \left\{ \|\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda \|\boldsymbol{\beta}\|_2^2 \right\} \tag{5}$$

donde $\|\boldsymbol{\beta}\|_2^2 = \sum_{j=1}^{p} \beta_j^2$ es la **norma L2** al cuadrado.

Nota: Típicamente NO penalizamos el intercepto β_0.

Ridge: Solución Analítica

A diferencia de Lasso, Ridge tiene solución cerrada:

$$\hat{\beta}^{Ridge} = (\mathbf{X}'\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}'\mathbf{Y} \tag{6}$$

Comparación con OLS:

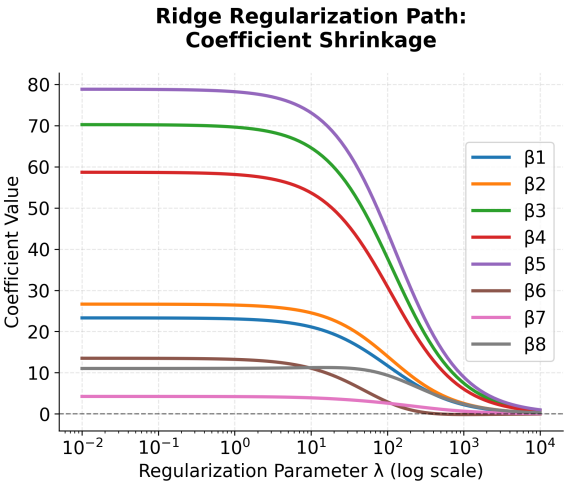
$$\hat{\beta}^{OLS} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y} \tag{7}$$

$$\hat{\beta}^{Ridge} = (\mathbf{X}'\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}'\mathbf{Y} \tag{8}$$

¿Qué hace $\lambda\mathbf{I}$?

- Añade λ a la diagonal de $\mathbf{X}'\mathbf{X}$
- Hace la matriz **siempre invertible** (incluso si $p > n$)
- "Encoge" los coeficientes hacia cero

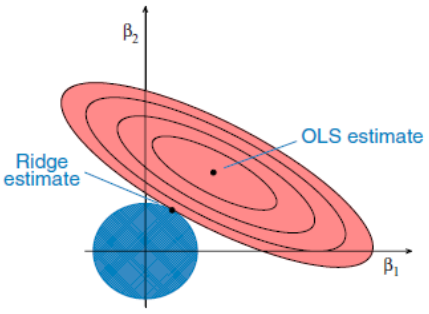
Ridge: Efecto de λ



Observaciones:

- Cuando $\lambda = 0$: coeficientes = OLS
- Cuando $\lambda \to \infty$: todos los coeficientes $\to 0$
- Los coeficientes se **encogen** pero nunca llegan a ser exactamente cero

Ridge: Interpretación Geométrica



Formulación equivalente (restringida):

$$\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} (Y_i - \mathbf{X}_i'\boldsymbol{\beta})^2 \quad \text{sujeto a} \quad \sum_{j=1}^{p} \beta_j^2 \leq t \tag{9}$$

El círculo L2 tiene **bordes suaves** $\rightarrow$ el óptimo rara vez cae en los ejes.

Lasso: Definición

Función objetivo de Lasso:

$$\hat{\boldsymbol{\beta}}^{Lasso} = \underset{\boldsymbol{\beta}}{\text{arg mín}} \left\{ \sum_{i=1}^{n} (Y_i - \mathbf{X}_i' \boldsymbol{\beta})^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\} \tag{10}$$

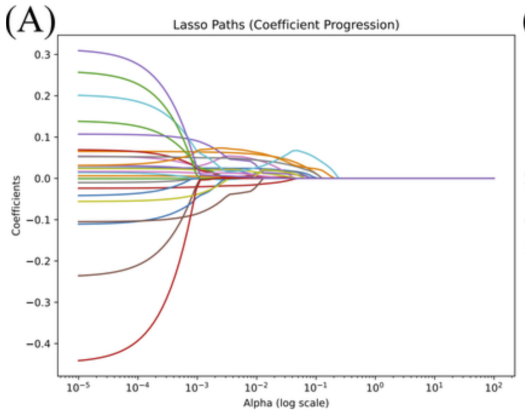
En notación matricial:

$$\hat{\boldsymbol{\beta}}^{Lasso} = \underset{\boldsymbol{\beta}}{\text{arg mín}} \left\{ \|\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda \|\boldsymbol{\beta}\|_1 \right\} \tag{11}$$

donde $\|\boldsymbol{\beta}\|_1 = \sum_{j=1}^{p} |\beta_j|$ es la **norma L1**.

LASSO = *Least Absolute Shrinkage and Selection Operator*

Lasso: La Magia de la Selección de Variables



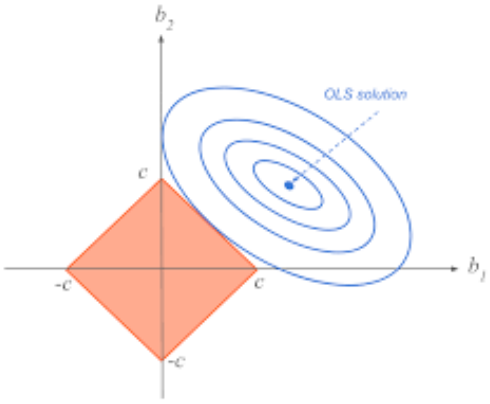
Trayectoria de coeficientes al variar λ

La diferencia crucial con Ridge

Lasso puede hacer coeficientes **exactamente cero**

- Ridge: encoge coeficientes, pero no elimina variables
- Lasso: fuerza a muchos coeficientes a ser 0
- Resultado: modelo más **simple** e **interpretable**

Lasso: Interpretación Geométrica



Contornos del RSS vs restricción L_1

Formulación equivalente (restringida):

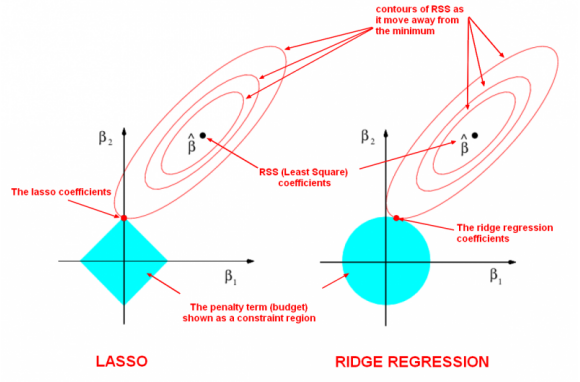
$$\min_{\beta} \sum_{i=1}^{n} (Y_i - \mathbf{X}_i'\beta)^2 \quad \text{sujeto a} \quad \sum_{j=1}^{p} |\beta_j| \leq t \tag{12}$$

Intuición clave

El rombo L_1 tiene **esquinas**

$\Rightarrow$ el óptimo suele caer en un eje $\Rightarrow$ $\beta_j = 0$

¿Por qué L1 produce ceros y L2 no?



Ridge (L2)

- Restricción circular
- Bordes suaves
- El óptimo puede caer en cualquier punto
- Coeficientes ≈ 0, pero casi nunca exactamente 0

Lasso (L1)

- Restricción romboidal
- Esquinas puntiagudas
- La elipse suele tocar una esquina
- Coeficientes exactamente 0 (selección de variables)

Lasso como Feature Selection

¿Por qué es útil que algunos $\hat{\beta}_j = 0$?

1. **Interpretabilidad:** Modelo más simple, más fácil de explicar
2. **Parsimonia:** Identifica las variables "realmente importantes"
3. **Reducción de dimensionalidad:** Útil cuando p es grande
4. **Costo de datos:** Menos variables = menos datos a recolectar

Advertencia

Si hay variables muy correlacionadas, Lasso elige una arbitrariamente y descarta las demás. Para correlación alta, considerar Elastic Net.

Elastic Net: Lo mejor de ambos mundos (tibieza)

Elastic Net combina L1 y L2:

$$\hat{\boldsymbol{\beta}}^{EN} = \arg\min_{\boldsymbol{\beta}} \left\{ \|\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda_1 \|\boldsymbol{\beta}\|_1 + \lambda_2 \|\boldsymbol{\beta}\|_2^2 \right\} \tag{13}$$

Formulación alternativa (más común en software):

$$\hat{\boldsymbol{\beta}}^{EN} = \arg\min_{\boldsymbol{\beta}} \left\{ \|\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda \left[\alpha \|\boldsymbol{\beta}\|_1 + (1-\alpha) \|\boldsymbol{\beta}\|_2^2 \right] \right\} \tag{14}$$

donde:

- $\lambda \geq 0$: intensidad total de regularización
- $\alpha \in [0, 1]$: mezcla entre L1 y L2
  - $\alpha = 1$: Lasso puro
  - $\alpha = 0$: Ridge puro

¿Cuándo usar Elastic Net?

Elastic Net es útil cuando:

- Hay **grupos de variables correlacionadas**
  - Lasso elegiría solo una del grupo
  - Elastic Net tiende a incluir o excluir el grupo completo
- $p >> n$ (más variables que observaciones)
  - Lasso selecciona a lo sumo n variables
  - Elastic Net no tiene esta limitación
- Queremos **selección de variables** pero con más estabilidad que Lasso

Nota práctica

Elastic Net tiene **dos hiperparámetros**: λ y α. Esto complica el tuning pero da más flexibilidad.

Resumen: Ridge vs Lasso vs Elastic Net

	Ridge	**Lasso**	**Elastic Net**
Penalización	$\|\boldsymbol{\beta}\|_2^2$	$\|\boldsymbol{\beta}\|_1$	$\alpha\|\boldsymbol{\beta}\|_1 + (1-\alpha)\|\boldsymbol{\beta}\|_2^2$
Coef. $= 0$	No	Sí	Sí
Selección var.	No	Sí	Sí
Sol. cerrada	Sí	No	No
Correl. altas	Estable	Inestable	Estable
Hiperparáms.	1 (λ)	1 (λ)	2 (λ, α)

Regla práctica:

- ¿Quieres selección de variables? $\rightarrow$ Lasso o Elastic Net
- ¿Hay multicolinealidad? $\rightarrow$ Ridge o Elastic Net
- ¿No sabes? $\rightarrow$ Prueba los tres y compara con CV

El Problema del Tuning

Recordemos: λ controla cuánta regularización aplicamos.

- λ muy pequeño $\rightarrow$ poco encogimiento $\rightarrow$ overfitting
- λ muy grande $\rightarrow$ mucho encogimiento $\rightarrow$ underfitting

¿Cómo encontrar el λ óptimo?

NO podemos usar el error de training

El error de training siempre favorece $\lambda = 0$ (OLS).

Solución: Cross-Validation

Elegimos λ que minimiza el **error de validación cruzada**.

Grid Search + Cross-Validation

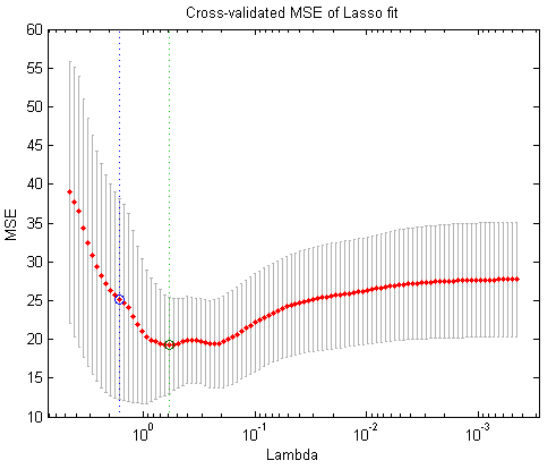
Procedimiento:

1. Definir una **grilla** de valores de λ: $\{\lambda_1, \lambda_2, \ldots, \lambda_m\}$
2. Para cada λ_j en la grilla:
   - Calcular $CV_{(K)}(\lambda_j)$ usando K-fold cross-validation
3. Seleccionar $\lambda^* = \arg\min_{\lambda} CV_{(K)}(\lambda)$
4. Re-entrenar el modelo con λ^* usando **todos** los datos de training

Nota: Típicamente usamos una grilla en escala **logarítmica**:

$$\lambda \in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 100, \ldots\}$$

Visualización del Tuning



Criterios de selección:

- λ_{min}: minimiza el error promedio de cross-validation.
- λ_{1se}: mayor λ dentro de 1 desviación estándar del mínimo.

Interpretación económica:

- A mayor $\lambda \rightarrow$ mayor penalización.
- Más coeficientes en cero $\rightarrow$ modelo más simple.
- Menor varianza, mayor sesgo.

La Regla de 1 Desviación Estándar

¿Por qué elegir λ_{1se} en lugar de λ_{min}?

- El error de CV tiene **variabilidad** (estimamos con una muestra finita)
- Si dos λ tienen CV error "estadísticamente igual", preferimos el **más simple**
- λ_{1se} es el λ más grande cuyo error está dentro de 1 SE del mínimo

Principio de parsimonia

Entre modelos con performance similar, elegimos el más simple. Esto reduce varianza y mejora interpretabilidad.

Importancia de Estandarizar

Antes de aplicar regularización: ¡ESTANDARIZAR!

Ridge y Lasso penalizan los coeficientes según su magnitud. Si las variables están en escalas diferentes, la penalización es injusta.

Ejemplo:

- X_1: ingreso en millones de pesos $\rightarrow$ β_1 pequeño
- X_2: edad en años $\rightarrow$ β_2 grande

Sin estandarizar, Lasso penalizaría más a β_2 simplemente por la escala.

Procedimiento correcto:

1. Dividir en train/test
2. Estandarizar con media y std de **training**
3. Aplicar regularización
4. (Opcional) Transformar coeficientes de vuelta a escala original

scikit-learn

Lo que haremos en el ejercicio práctico:

1. Cargar y explorar los datos
2. Dividir en train/test
3. Estandarizar (fit en train, transform en ambos)
4. Ajustar Ridge, Lasso y Elastic Net
5. Tuning de λ con cross-validation
6. Comparar modelos con MSE de test
7. Analizar qué variables seleccionó Lasso

Funciones clave

`RidgeCV`, `LassoCV`, `ElasticNetCV`, `StandardScaler`

Resumen: Ideas Clave

1. **Regularización:** Introducir sesgo para reducir varianza
2. **Ridge (L2):** Encoge coeficientes, nunca a cero exactamente
3. **Lasso (L1):** Encoge y puede hacer coeficientes exactamente cero $\rightarrow$ selección de variables
4. **Elastic Net:** Combina L1 y L2, útil con correlaciones altas
5. **Hiperparámetros:** Se eligen con cross-validation, no con error de training
6. **Estandarizar:** Siempre antes de regularizar, con parámetros de training

Referencias y Lecturas Recomendadas

Textos fundamentales:

- James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). *An Introduction to Statistical Learning* (2nd ed.). Capítulo 6. [Gratis online]
- Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning* (2nd ed.). Capítulo 3. [Gratis online]

Documentación de scikit-learn:

- Linear Models in scikit-learn

Paper original de Lasso:

- Tibshirani, R. (1996). Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society B*, 58(1), 267-288.

¡Gracias!

s.neira10@uniandes.edu.co