

Fundamentos de ciencia de datos con R

Gema Fernández-Avilés y José-María Montero

2023-06-05

Índice general

Prefacio	5
¡Hola mundo!	5
¿Por qué este libro?	6
¿A quién va dirigido?	7
El paquete CDR	8
¿Por qué R?	8
Agradecimientos	9
1. Modelos <i>sparse</i> y métodos penalizados de regresión	11
1.1. Introducción	11
1.2. Selección del mejor subconjunto	12
1.3. Métodos <i>shrinkage</i>	18

Prefacio

¡Hola mundo!

El siglo XXI está siendo testigo de grandes cambios vertiginosos en el contexto social y tecnológico, entre otros. Los tiempos han cambiado, la sociedad se ha globalizado y “exige” respuestas inmediatas a problemas muy complejos. Vivimos en el mundo de la **información**, de los **datos**, o mejor, de las **bases de datos masivas**, y los ciudadanos y, sobre todo, las empresas y los gobiernos, dirigen su mirada hacia el mundo científico para que les ayude a “**oír las historias**” que cuentan esos datos acerca de la realidad de la que han sido extraídos. Y dado su enorme volumen y sofisticación (en el nuevo mundo las imágenes y los textos, por ejemplo, también son datos), exigen algoritmos de nueva generación en el campo del *machine learning*, o incluso del *deep learning*, para “oír las historias” que cuentan. No parecen mirar al “antiguo” investigador científico, sino al “nuevo” *científico de datos*.

Ello, inevitablemente, se traduce en la necesidad de profesionales con una gran capacidad de adaptación a este nuevo paradigma: los científicos de datos, también llamados por algunos los “nuevos hombres del Renacimiento”, para lo cual las Universidades y demás instituciones educativas especializadas se apresuran a incluir el grado de Ciencia de Datos en su oferta educativa y a ofrecer seminarios de software estadístico de acceso abierto para sus estudiantes de primeros cursos.

Con la emergencia de la nueva sociedad, en la que el manejo de la ingente cantidad de información que genera se hace absolutamente necesario para circular por ella, la **Ciencia de Datos** ha venido para quedarse. Sin embargo, el mundo de la Ciencia de Datos es cualquier cosa menos sencillo. En él, cualquier ayuda, cualquier guía es bienvenida. Por ello, es muy recomendable que la persona que se quiera introducir en él, sea con fines de investigación o con fines profesionales, se agarre de la mano de un guía especializado que le lleve, de una manera amena, comprensible y eficiente, desde el planteamiento de su problema y la captura de la información necesaria para poderle dar una solución, hasta la redacción de las conclusiones finales que ha obtenido con los modernos informes reproducibles colaborativos. Y como en la parte central de ese camino tendrá que luchar con grandes gigantes (en la actualidad denominados técnicas estadísticas y algoritmos), el guía tendrá que explicarle, de manera sencilla y amena, en qué consiste la lucha (las técnicas y los algoritmos) y cómo llegar a la victoria lo más rápido posible, enseñándole a moverse por el mundo del software estadístico, en nuestro caso **R**, que le permitirá realizar los cálculos necesarios para vencer al problema planteado a una velocidad vertiginosa.

En resumen, la información masiva y el moderno tratamiento estadístico de la misma son la “mano invisible” que gobierna la sociedad del siglo XXI, y este manual pretende ser el guía anteriormente mencionado que le llevará de la mano cuando quiera caminar por ella.

¿Por qué este libro?

Lo dicho anteriormente ya justifica por sí solo la aparición de este manual. Afortunadamente, no es el primero en la materia, pues son ya bastantes los materiales de calidad publicados sobre Ciencia de Datos. Sin embargo, quizás, éste pueda ser considerado el más completo. Y ello por varias razones.

La primera es su **completitud**: este manual lleva de la mano al lector desde el planteamiento del problema hasta el informe que contiene la solución al mismo; o desde no saber qué hacer con la información de la que dispone, hasta ser capaz de transformar tales bases de datos masivas, y casi imposibles de manejar, en respuestas a problemas fundamentales de una empresa, institución o cualquier agente social.

La segunda es su **amplitud temática**:

- (I) Parte de las dos primeras preguntas que un neófito se puede hacer sobre esta temática: ¿qué es eso de la Ciencia de Datos que está en boca de todos? Y, ¿qué diablos es **R** y cómo funciona?
- (II) Enseña cómo moverse en la jungla del *Big Data* y de los “nuevos” tipos de datos, siempre bajo el paraguas de la ética de los datos y del buen gobierno de dichos datos.
- (III) Muestra al lector cómo obtener conocimiento de la oscuridad del enorme banco de información a su disposición, que no sabe cómo abordar ni manejar.
- (IV) No deja a nadie atrás, y de forma previa al contenido central del manual (las técnicas de Ciencia de Datos), incluye unas breves, pero magníficas, secciones sobre los rudimentos de la probabilidad, la inferencia estadística y el muestreo, para aquéllos no familiarizados con estas cuestiones.
- (V) Aborda una treintena de técnicas de Ciencia de Datos en el ámbito de la modelización, análisis de datos cualitativos, discriminación, *machine learning* supervisado y no supervisado, con especial incidencia en las tareas de clasificación y clusterización -así como, en el caso no supervisado, de reducción de la dimensionalidad, escalamiento multidimensional y análisis de correspondencias-, *deep learning*, análisis de datos textuales y de redes, y, finalmente, ciencia de datos espaciales (desde las perspectivas de la geoestadística, la econometría espacial y los procesos de punto).
- (VI) Hace especial hincapié en la reproducibilidad en tiempo real (o no) entre los distintos miembros de un equipo (sea universitario, empresarial, o del tipo que sea) y en la difusión de los resultados obtenidos, enseñando al lector cómo generar informes reproducibles mediante RMarkdown y documentos Quarto o en otros modernos formatos.
- (VII) Dedica un capítulo a la creación de aplicaciones web interactivas (con Shiny).

Índice general

7

- (viii) Para aquéllos con pasión por la codificación, y que quieran compartir código y colaborar con otros desarrolladores, este manual aborda la gestión rápida y eficaz de proyectos (del tamaño que sean) mediante Git, un sistema de control de versiones distribuido, gratuito y de código abierto, y GitHub, un servicio de alojamiento de repositorios Git del cual, aquellos no familiarizados con la cuestión de la codificación, o con aversión a ella, podrán tomar el código que necesitan.
- (ix) Muestra al lector los primeros pasos para iniciarse en el geoprocесamiento en la nube.
- (x) Y, finalmente, aborda más de una docena de casos de uso (en medicina, periodismo, economía, criminología, marketing, moda, demanda de electricidad, cambio climático, reconocimiento de patrones en la forma de tuitear...) que ilustran la puesta en práctica de todos los conocimientos anteriormente adquiridos.

La cuarta razón es que todo lo que el lector aprende en este manual lo puede reproducir y poner en práctica inmediatamente con **R**, puesto que el manual está trufado de *chunks* (o trozos de código **R**) que no tiene más que cortar y pegar para reproducir los ejemplos que se muestran en el libro, cuyos datos están en el paquete CDR; o utilizar dichas *chunks* para abordar el problema que le ocupa con los datos que tenga a su disposición. Una buena razón, sin duda. Por consiguiente, el manual es una buena combinación “teoría-práctica-software” que permite abordar cualquier problema que el científico de datos se plante en cualquier disciplina o situación empresarial, médica, periodística...

La quinta es su **variedad de perspectivas**. Son **más de 40 los participantes** en este manual. Algunos de ellos, prestigiosos profesores universitarios; otros, destacados miembros de instituciones públicas; otros, CEOs de empresas en la órbita de la ciencia de datos; otros, *big names* del mundo de **R** software... El manual es, sin duda, un magnífico ejemplo de colaboración Universidad-Empresa para buscar soluciones a los problemas de las sociedades modernas.

¿A quién va dirigido?

Fundamentos de ciencia de datos con R está dirigido a todos aquellos que desean desarrollar las habilidades necesarias para abordar proyectos complejos de Ciencia de Datos y “pensar con datos” (como lo acuñó Diane Lambert, de Google). El deseo de resolver problemas utilizando datos es su piedra angular. Por tanto, como se avanzó anteriormente, este manual no deja a nadie atrás, y lo único que requiere es “el deseo de resolver problemas utilizando datos”. No excluye ninguna disciplina, no excluye a las personas que no tengan un elevado nivel de análisis estadístico de datos, no excluye a nadie. Se ha procurado una combinación de rigor y sencillez, y de teoría y práctica, todo ello con sus correspondientes códigos en **R**, que satisfaga tanto a los más exigentes como a los principiantes.

También está destinado a todos aquellos que quieran sustituir la navegación por la web (la búsqueda del video, publicación de blog o tutorial *online* que solucione su problema –frustración tras frustración por la falta de consistencia, rigor e integridad de dichos materiales, así como por su sesgo hacia paquetes singulares para la implementación de las cuestiones que tratan–), por

una “**biblia de la ciencia de datos**” rigurosa pero sencilla, práctica y de aplicación inmediata sin ser ni un experto estadístico ni un experto informático.

Pero si a alguien está destinado especialmente, es a la comunidad hispano hablante. Este manual es un guiño a dicha comunidad, para que tenga a su disposición, en su lengua nativa, uno de los mejores manuales de Ciencia de Datos de la actualidad.

El paquete CDR



El paquete **CDR** contiene la mayoría de conjuntos de datos utilizados en este libro que no están disponibles en otros paquetes. Para instalarlo use la función `install_github()` del paquete `remotes`.

```
# este comando sólo necesita ser ejecutado una vez
# si el paquete remotes no está instalado, descomentar para instalarlo

# install.packages("remotes")
remotes::install_github("cdr-book/CDR")
```

La lista de todos los conjuntos de datos puede obtenerse haciendo `data()`.

```
library('CDR')
data(package = "CDR")
```

Este paquete ayudará al lector a reproducir todos los ejemplos del libro. De acuerdo con las mejores prácticas en **R**, el paquete **CDR** sólo contiene los datos utilizados en el libro.

¿Por qué R?

R es un lenguaje de código abierto para computación estadística que se ha consolidado entre la comunidad científica internacional, en las últimas dos décadas, como una herramienta de primer

Índice general

9

nivel, consolidándose como líder permanente en el ámbito de la implementación de metodologías estadísticas para el análisis de datos. La utilidad de **R** para la Ciencia de Datos deriva de un fantástico ecosistema de paquetes (activo y en crecimiento), así como de un buen elenco de otros excelentes recursos: libros, manuales, *blogs*, foros y *chats* interactivos en las redes sociales, y una gran comunidad dispuesta a colaborar, a orientar y a resolver diferentes cuestiones relacionadas con **R**.

Por otra parte, **R** es el lenguaje estadístico y de análisis de datos más utilizado en la mayoría de los entornos académicos y, cómo no, por una larga lista de importantes empresas, entre las que se cuentan Facebook (análisis de patrones de comportamientos relacionado con actualizaciones de estado e imágenes de perfil), Google (para la efectividad de la publicidad y la previsión económica), Twitter (visualización de datos y agrupación semántica), Microsoft (adquirió la empresa Revolution R), Uber (análisis estadístico), Airbnb (ciencia de datos), IBM (se unió al grupo del consorcio R), New York Times (visualización)...

La comunidad **R** también es particularmente generosa e inclusiva, y hay grupos increíbles, como *R-Ladies* y *Minority R Users*, diseñados para ayudar a garantizar que todos aprendan y usen las capacidades de **R**.

Agradecimientos

No queremos dar por finalizado este prefacio sin agradecer a los 44 autores participantes en esta obra su esfuerzo por condensar, en no más de 20 páginas, la teoría, práctica y tratamiento informático de la parte de la Ciencia de Datos que les fue encargada. Y no sólo eso; el “más difícil todavía” fue que debían dirigirse a un abanico de potenciales lectores tan grande como personas haya con “el deseo de resolver problemas utilizando datos”. Era misión imposible. Sin embargo, a la vista del resultado, ha sido misión cumplida. El esfuerzo mereció la pena.

Además, nos gustaría agradecer el apoyo incondicional recibido por (en orden alfabético): Itzcoatl Bueno, Ismael Caballero, Emilio L. Cano, Diego Henangómez, Ricardo Pérez, Manuel Vargas y Jorge Velasco.

También queremos poner de manifiesto que la edición de este texto ha sido financiada por diversos entes de la Universidad de Castilla-La Mancha. En su mayor parte, por el **Máster en Data Science y Business Analytics (con R software)** (a través de la orgánica: 02040M0280), pero también por la Facultad de Ciencias Jurídicas y Sociales de Toledo (a través de su contrato programa: orgánica 00440710), el Departamento de Economía Aplicada I (mediante sus fondos departamentales, DEAI 00421I126) y el Grupo de Investigación Economía Aplicada y Métodos Cuantitativos (que ha dedicado parte de sus fondos a la edición de esta obra, orgánica 01110G3044-2023-GRIN-34336).

A todos, eternamente agradecidos por ayudarnos en este reto de transformar la oscuridad en conocimiento, de convertir en una ciencia y en un arte la difícil tarea de sacar valor de los datos, el petróleo del futuro. Quizás en este momento no seamos conscientes de que hemos puesto nuestro granito de arena a la ciencia que, a buen seguro, juegue uno de los papeles más importantes de este siglo, caracterizado por el predominio de la información. Una ciencia, la Ciencia de Datos, que combina el análisis estadístico de datos, la algoritmia y el conocimiento del

negocio para sacar valor del bien más abundante de la sociedad en la que vivimos: la información. Una disciplina cuyo dominio caracteriza a los científicos de datos (también denominados los nuevos personajes del Renacimiento), profesión que ya fue calificada hace más de veinte años en la *Harvard Business Review* y en *The New York Times*, entre otros, como la “más sexy del siglo XXI”.

Nota

Este manual está publicado por [McGraw Hill](#). Las copias físicas están disponibles en [McGraw Hill](#). La versión *online* se puede leer de forma gratuita en <https://cdr-book.github.io/> y tiene la [licencia de Creative Commons Reconocimiento-NoComercial-SinObraDerivada 4.0 Internacional](#).

Si tiene algún comentario o sugerencia, no dude en contactar con los editores y los autores. ¡Gracias!

Capítulo 1

Modelos *sparse* y métodos penalizados de regresión

María Durbán

Universidad Carlos III de Madrid

1.1. Introducción

El modelo de regresión lineal múltiple: $y = \beta_0 + \beta_1 X_1 + \dots + \beta_p X_p + \varepsilon$, visto en el Cap. ??, a pesar de su simplicidad, tiene importantes ventajas como la **interpretabilidad** y su buen poder **predictivo** en muchas situaciones.

En este capítulo enseña cómo se puede hacer el modelo aún más interpretable y mejor predictor, y para conseguirlo se reemplaza el método de estimación de mínimos cuadrados por un método alternativo. Más concretamente, el objetivo de este capítulo es presentar técnicas para mejorar la:

- **Precisión de la predicción:** en particular, cuando el número de variables es mayor que el número de observaciones: $p > n$ (algo que ocurre con mucha frecuencia hoy en día). En este caso no se pueden utilizar mínimos cuadrados ya que la matriz de diseño no es de rango completo y, por lo tanto, no se puede encontrar una solución única al problema de minimización. Por ello, se necesita reducir el número de variables, que además, evitará que se sobreajusten los datos.
- **Interpretabilidad del modelo:** al eliminar las variables irrelevantes (es decir, haciendo cero los correspondientes coeficientes) se obtendrá un modelo más fácil de interpretar.

En base a lo anterior,a continuación se presentan varios métodos para llevar a cabo de forma automática la reducción de variables en el modelo, actividad también denominada **selección de variables**. Tales métodos son:

- **Selección del mejor subconjunto:** su objetivo es identificar el subconjunto de $k < p$ predictores que contenga sólo los que mejor expliquen el comportamiento de la variable respuesta.
- **Shrinkage:** en este caso no se quieren seleccionar variables explícitamente, sino que se añade una penalización que penaliza el número de coeficientes o su tamaño.
- **Reducción de la dimensión:** el objetivo es proyectar los p -predictores en un subespacio de dimensión más pequeña (mediante el uso de combinaciones lineales de las variables predictoras, las cuales se usarán como “nuevos” predictores). Dichas combinaciones lineales se llaman **componentes principales** y a su análisis se dedica el Cap. ??.

En este Capítulo se ven los dos primeros métodos. Para el tercero, se remite al lector al Cap. ??.

1.2. Selección del mejor subconjunto

Supóngase que se tiene acceso a p variables predictoras, pero se quiere un modelo más simple que involucre sólo a un subconjunto de esos p predictores. La forma lógica de conseguirlo es considerar todos los posibles subconjuntos de los p predictores y elegir el mejor modelo de entre todos los modelos construidos con cada uno de los subconjuntos de variables. Los pasos a seguir serían:

1. Se crea el modelo **nulo**, M_0 , que es aquel que únicamente contiene la ordenada en el origen y ningún predictor. Este modelo simplemente predice la media muestral para cada observación.
2. Para cada valor de $k = 1, 2, \dots, p$, se calculan los $\binom{p}{k}$ modelos que contienen k predictores. Es decir, los p modelos que contienen 1 predictor, los $p \times (p - 1)/2$ modelos que contienen 2 predictores, etc.
3. Para cada valor de k , se elige el mejor entre los $\binom{p}{k} = \frac{p!}{(p-k)!k!}$ posibles modelos y se denota por M_k . Es decir, M_1 sería el mejor modelo entre los p modelos con una única variable, M_2 sería el mejor modelo entre los modelos con dos variables, etc. En este caso, el **mejor** modelo sería aquel cuyo RSS (suma de residuos al cuadrado) sea menor, o equivalentemente, aquel cuyo R^2 sea mayor.
4. Finalmente, entre los modelos: M_1, \dots, M_p se elige el mejor utilizando un criterio como AIC (criterio de información de Akaike), BIC (criterio de información bayesiano) o R^2 ajustado.

Este método se puede usar también en el caso de GLMs, si bien, en este caso, se usa la *deviance* en vez de RSS .

1.2.1. Procedimiento con R: la función `regsubset()`

En esta subsección se aplica el método descrito al conjunto de datos `Hitters` del paquete `ISLR2`. El objetivo es predecir el sueldo, `Salary`, de jugadores de béisbol a partir de varias variables asociadas con su rendimiento el año anterior.

La variable `Salary` no está disponible para algunos de los jugadores. Éstos se pueden identificar con la función `is.na()`. La función `sum()` permite ver cuántos hay. Se utiliza `na.omit()` para eliminarlos.

```
library("ISLR2")
Hitters <- na.omit(Hitters)
```

La función `regsubsets()` del paquete `leaps` lleva a cabo la selección del mejor subconjunto de variables predictoras, identificando el mejor modelo que contiene un número dado de ellas (1,2,3, etc.) atendiendo a *RSS*. La sintaxis usada es similar a la de la función `lm()`.

```
library("leaps")
regfit_full <- regsubsets(Salary ~ ., Hitters)
```

Los resultados se pueden ver usando `summary()`, donde se muestra el mejor modelo para cada número específico de variables. Las variables incluidas en cada modelo se indican con un asterisco. Por ejemplo, el mejor modelo con dos variables incluye `Hits` y `CRBI`. Por defecto, `regsubsets()` solo muestra los resultados de los modelos que contienen hasta ocho variables. La opción `nvmax` se puede usar para incrementar esta cantidad, por ejemplo hasta 19 variables (que es el número de variables predictoras en el conjunto de datos):

```
regfit_full <- regsubsets(Salary ~ .,
  data = Hitters,
  nvmax = 19
)
reg_summary <- summary(regfit_full)
```

La función `summary()` devuelve diferentes medias de bondad de ajuste: R^2 , RSS , R^2 ajustado, C_p y BIC que se utilizan para elegir el *mejor* de entre todos los modelos.

```
names(reg_summary)
#> [1] "which"   "rsq"     "rss"     "adjr2"   "cp"      "bic"      "outmat"  "obj"
reg_summary$adjr2
#> [1] 0.3188503 0.4208024 0.4450753 0.4672734 0.4808971 0.4972001 0.5007849
#> [8] 0.5137083 0.5180572 0.5222606 0.5225706 0.5217245 0.5206736 0.5195431
#> [15] 0.5178661 0.5162219 0.5144464 0.5126097 0.5106270
```

En el ejemplo, el R^2 ajustado mayor corresponde al modelo con 11 variables.

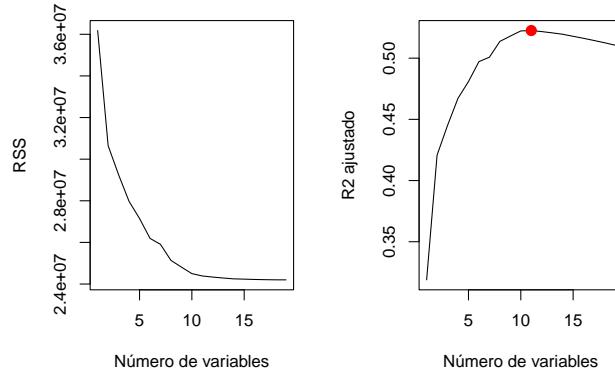


Figura 1.1: Valores de R^2 y R^2 ajustado correspondientes a modelos con distinto número de variables

Los resultados también se pueden mostrar y dibujar simultáneamente; por ejemplo, los valores de RSS y R^2 ajustado de todos los modelos se muestran en la Fig. 1.1.

Otra manera de visualizar los resultados es:

```
plot(regfit_full, scale = "adjr2")
```

La primera fila tiene un cuadrado negro en cada una de las variables explicativas del modelo con mayor R^2 ajustado (en este caso, sería similar para los otros criterios).

Varios modelos tienen un valor de R^2 ajustado próximo a 0,52, pero es el modelo con 11 variables el que alcanza el mayor valor. La función `coef()` permite ver los coeficientes estimados de este modelo.

```
coef(regfit_full, 11)
#> (Intercept)      AtBat       Hits       Walks      CATBat      CRUNS
#> 135.7512195 -2.1277482  6.9236994  5.6202755 -0.1389914  1.4553310
#> CRBI          CWalks     LeagueN   DivisionW    PutOuts      Assists
#>  0.7852528 -0.8228559 43.1116152 -111.1460252   0.2894087  0.2688277
```

1.2.2. Selección *stepwise*

Cuando el número de variables predictoras, p , es grande, el método anterior es computacionalmente muy costoso ya que el número de posibles combinaciones de variables crece de una manera alarmante. En general, la función `regsubset()` puede lidiar con hasta 30-40 variables predictoras. Además, otro problema es el sobreajuste. Si se tienen 40 variables, se estarían ajustando millones de modelos, y puede que el modelo elegido funcione muy bien en los datos

1.2. Selección del mejor subconjunto

15

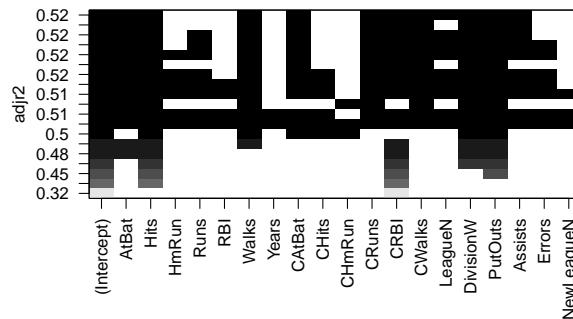


Figura 1.2: Variables seleccionadas en cada uno de los modelos y su correspondiente valor de R^2 ajustado.

utilizados para su construcción, pero no tan bien en un nuevo conjunto de datos. Una alternativa es el método **stepwise**. La idea detrás de este método es similar a la anterior, pero se busca el mejor modelo entre un conjunto mucho más pequeño de modelos.

Hay dos posibilidades de hacer stepwise: **forward** y **backward**. Ambas son bastante parecidas; la principal diferencia es el modelo del que se parte: del modelo sin ninguna variable predictora (*forward*) o del modelo con todas ellas (*backward*).

1.2.2.1. Forward stepwise

En este caso se comienza con el **modelo nulo**, M_0 , y se van añadiendo variables secuencialmente. En particular, en cada paso (*step*) la variable que proporciona la mayor mejora al ajuste es la que se añade al modelo. Los pasos a seguir serían:

1. Se crea el modelo nulo, M_0 .
2. Para cada valor de $k = 0, 1, 2, \dots, p$:
 - a) Se consideran todos los $p - k$ modelos que surgen de aumentar el modelo M_k con un predictor.
 - b) Se elige el **mejor** de esos $p - k$ modelos, que se denota M_{k+1} . El término **mejor** significa tener el *RSS* más bajo o el R^2 más alto.
3. Se elige el mejor de los modelos M_0, \dots, M_p en función de un criterio que tenga en cuenta la complejidad del modelo, como AIC, BIC o R^2 ajustado.

Este enfoque tiene ventajas computacionales claras, ya que el número de modelos ajustados es mucho menor, pero no garantiza que el modelo elegido sea el mejor modelo posible, especialmente si existe correlación entre las variables predictoras.

1.2.2.2. Backward stepwise

En este caso, se comienza con el modelo que incluye las p las variables predictoras y se van eliminando de forma iterativa hasta llegar al modelo nulo (M_0). Los pasos serían:

1. Se ajusta el modelo M_p , que contiene todas (p) las variable predictoras.
2. Para cada valor de $k = p, p-1, \dots, 1$: *a)* Se consideran todos los k modelos que surgen de reducir en el modelo M_k un predictor, es decir, modelos con $k-1$ variables predictoras.
b) Se elige el **mejor** de esos k modelos, que se denota M_{k-1} . Dicho modelo será el que tenga el RSS más bajo o el R^2 más alto.
3. Entre los modelos M_0, \dots, M_p se elige el mejor en función de un criterio como AIC, BIC o R^2 ajustado.

Tanto en el caso *forward* como en el caso *backward*, se busca el mejor modelo “sólo” entre $1+p(p+1)/2$ modelos, lo que los hace recomendables frente a la selección del mejor subconjunto de variables cuando p es demasiado grande..

El método *backward stepwise* necesita que el número de observaciones n sea mayor que el de variables predictoras p (ya que necesita ajustar el modelo con todas las variables). Por el contrario, el método *forward stepwise* se puede usar incluso cuando $n < p$.

1.2.2.3. Procedimiento con R: la función `regsubset()`

La función `regsubset()` permite utilizar los métodos *forward* y *backward*, usando los argumentos `method = "forward"` o `method = "backward"`:

```
regfit_fwd <- regsubsets(Salary ~ .,
  data = Hitters,
  nvmax = 19, method = "forward"
)
regfit_bwd <- regsubsets(Salary ~ .,
  data = Hitters,
  nvmax = 19, method = "backward"
)
```

Los métodos mejor subconjunto de variables, *forward stepwise* y *backward stepwise* no tienen por qué seleccionar el mismo (mejor) modelo. Ni siquiera tienen por qué seleccionar el mismo modelo para cada número de predictores en la fase previa a la selección final. Así ocurre, por ejemplo, cuando el número de variables predictoras es $k = 2$:

```
coef(regfit_full, 2)
#> (Intercept)      Hits       CRBI
#> -47.9559022   3.3008446  0.6898994
coef(regfit_fwd, 2)
#> (Intercept)      Hits       CRBI
```

1.2. Selección del mejor subconjunto

17

```
#> -47.9559022  3.3008446  0.6898994
coef(regfit_bwd, 2)
#> (Intercept)      Hits       CRuns
#> -50.8174029   3.2257212  0.6614168
```

En la etapa final de selección, el modelo seleccionado no tiene por qué ser el mismo en función de los distintos criterios de selección (aunque normalmente lo es). Lo habitual es decidir un criterio para elegir el mejor modelo (R^2 ajustado, BIC , etc.) y seleccionarlo en función de él. En el ejemplo, seleccionando el criterio del R^2 ajustado, el mejor modelo es el que tiene 11 variables, tanto con el criterio *forward* como con el *backward*¹:

```
which.max(summary(regfit_fwd)$adjr2)
#> [1] 11
which.max(summary(regfit_bwd)$adjr2)
#> [1] 11
```

Con el criterio BIC también se selecciona el modelo con 11 variables predictoras:

```
which.min(summary(regfit_fwd)$bic)
#> [1] 6
which.min(summary(regfit_bwd)$bic)
#> [1] 8
```

Otra posibilidad es utilizar como criterio de selección el error de predicción, y para ello se puede echar mano de algún esquema de validación cruzada. A continuación se ilustra el caso en el que se divide la muestra en dos subconjuntos: *training* y *testing*, pero se puede utilizar cualquier otro método (validación cruzada k-grupos*, etc. Véase Sec. ??).

```
set.seed(1)
entreno <- sample(c(TRUE, FALSE), nrow(Hitters), replace = TRUE)
test <- (!entreno)
```

Se utiliza `regsubsets()` en la muestra de entrenamiento para obtener los modelos con distinto número de variables predictoras:

```
regfit_best <- regsubsets(Salary ~ ., data = Hitters[entreno, ], nvmax = 19)
```

Para calcular el error de predicción, dado que la función `regsubset()` no tiene asociada una función `predict()`, se han de calcular “manualmente” los valores predichos para la muestra de test. Para eso se necesita la matriz de diseño del modelo.

¹Recuérdese que con el método del mejor subconjunto de variables predictoras el criterio del R^2 ajustado también seleccionó el modelo con 11 variables.

```
test.mat <- model.matrix(Salary ~ ., data = Hitters[test, ])
```

Ahora, para cada modelo de tamaño k , se extraen los coeficientes de `regfit_best` para el mejor modelo de ese tamaño, se multiplica el vector de coeficientes por la matriz de diseño y se obtienen las predicciones; a continuación se calcula el error cuadrático medio (MSE).

```
val_errors <- rep(NA, 19)
for (i in 1:19) {
  coefi <- coef(regfit_best, id = i)
  pred <- test.mat[, names(coefi)] %*% coefi
  val_errors[i] <- mean((Hitters$Salary[test] - pred)^2)
}
```

Con este criterio, el mejor modelo es el que contiene 7 variables:

```
val_errors
#> [1] 164377.3 144405.5 152175.7 145198.4 137902.1 139175.7 126849.0 136191.4
#> [9] 132889.6 135434.9 136963.3 140694.9 140690.9 141951.2 141508.2 142164.4
#> [17] 141767.4 142339.6 142238.2
```

1.3. Métodos *shrinkage*

Los métodos anteriores se basan en el ajuste de modelos mediante mínimos cuadrados ordinarios. Los métodos *shrinkage*, sin embargo, se basan en una modificación del procedimiento de mínimos cuadrados ordinarios que consiste en añadir una penalización que *encoje* los coeficientes del modelo (normalmente hacia 0). Una de las ventajas de este tipo de métodos es que reduce la varianza de los coeficientes estimados.

Recuérdese que en el ajuste por mínimos cuadrados las estimaciones de $\beta_0, \beta_1, \dots, \beta_p$ son los valores que minimizan:

$$RSS = \sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2.$$

1.3.1. Regresión *ridge*

La **regresión ridge**² añade un término de penalización controlado por un parámetro (que habrá que elegir) que penalizará la magnitud de los coeficientes. Cuanto más grande es el coeficiente mayor es la penalización. En consecuencia, en la regresión *ridge* la expresión que se minimiza

²La traducción en español sería regresión contraída o regresión alomada.

1.3. Métodos shrinkage

19

para obtener las estimaciones de los parámetros del modelo es:

$$\sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^p \beta_j^2 = RSS + \lambda \sum_{j=1}^p \beta_j^2. \quad (1.1)$$

En realidad, lo que se está haciendo es hacer pagar al modelo un precio (en términos de ajuste) por el hecho de que los coeficientes no sean cero, y el precio será tanto mayor cuanto más grande sea la magnitud del coeficiente. A esta penalización se le llama **penalización shrinkage** porque “anima” a los coeficientes a que se *contraigan* hacia 0 (así es como este método favorece la simplicidad de los modelos). La magnitud de dicha contracción está gobernada por lambda, el parametro de afinado o regulación (también conocido en la jerga como “de tuneado”). Si $\lambda = 0$, se está en el caso de mínimos cuadrados ordinarios, y cuanto mayor sea λ , mayor será el precio a pagar para que esos coeficientes sean distintos de 0. Si λ es extremadamente grande, los coeficientes estarán muy próximos a 0, para que el segundo término pequeño (recuérdese que se minimiza RSS más la penalización). Aunque valores más grandes de los coeficientes proporcionasen un mejor ajuste (y por lo tanto un menor RSS), el término de penalización aumentaría se hará grande y no se alcanzaría el mínimo. Por lo tanto λ gobierna el equilibrio entre un buen ajuste del modelo y el tamaño de los coeficientes (y por lo tanto el número de coeficientes distintos de cero).

La elección del valor de λ es un punto crucial de este tipo de regresión. Para su determinación se suelen utilizar procedimientos de validación cruzada.

1.3.1.1. Escalado de variables predictoras

Un punto importante en regresión ridge es si las variables predictoras están escaladas o no.

El método de mínimos cuadrados ordinarios es *invariante a la escala (scale-invariant)*, es decir, que si se multiplica una variable predictora X_j por una constante c , el coeficiente estimado se multiplicada por $1/c$, pero $X_j \hat{\beta}_j$ no cambia. Sin embargo, en el caso de la regresión *ridge* los coeficientes estimados pueden cambiar sustancialmente ante un cambio de escala (es decir, si se multiplica una variable predictora por una constante), ya que todos los coeficientes forman parte del término de penalización. Por lo tanto, antes de utilizar la regresión *ridge* (o cualquier método de regularización) es importante **estandarizar las variables predictoras**, dividiendo cada variable por su desviación estándar, de forma que todas tengan desviación estándar igual a 1:

$$\tilde{x}_{ij} = \frac{x_{ij}}{\sqrt{\frac{1}{N} \sum_{i=1}^N (x_{ij} - \bar{x}_{ij})^2}}.$$

Con esto se consigue que los coeficientes estén en “igualdad de condiciones”.

En muchas ocasiones la regresión *ridge* da lugar a un menor MSE que el obtenido con mínimos cuadrados ordinarios. Sin embargo, por muy grande que sea λ los coeficientes no serán 0, sino que estarán próximos a cero, por lo que **este método no es realmente un método de selección de variables**.

Sin embargo, la regresión ridge puede ser muy útil cuando hay variables predictoras altamente correlacionadas pero se desea mantener todas en el modelo. En estos casos, la regresión ridge soluciona los problemas de multicolinealidad.

1.3.1.2. Procedimiento con R: la función `glmnet()`

Para llevar a cabo la regresión *ridge* (y para otros métodos de regresión *shrinkage*) se usa el paquete `glmnet`. La función principal en este paquete se llama también `glmnet()`. Esta función tiene una sintaxis un poco diferente a las funciones usuales para el ajuste de distintos modelos en **R**. Es necesario pasarle la matriz \mathbf{X} de variables predictoras (sin la columna correspondiente a la ordenada en el origen) y el vector \mathbf{y} con la variable respuesta. Para ilustrar su uso se utilizan los datos anteriores sobre béisbol.

```
x <- model.matrix(Salary ~ ., Hitters)[, -1]
y <- Hitters$Salary
```

La función `glmnet()` tiene un argumento, `alpha`, que determina el tipo de penalización que se añade en el modelo. En el caso de regresión *ridge*, `alpha=0`.

Por defecto, la función `glmnet()` elige de forma automática el rango de valores de λ . Sin embargo, a modo ilustrativo, se va a elegir la rejilla de valores que van desde $\lambda = 10^{10}$ hasta $\lambda = 10^{-2}$, cubriendo de esta forma una gran gama de escenarios, desde el modelo nulo (solo la ordenada en el origen) hasta el caso de mínimos cuadrados ordinarios. Más adelante se verá que se puede llevar a cabo el ajuste del modelo para un valor determinado de λ que no esté entre los de la rejilla inicial.

```
library("glmnet")
grid <- 10^seq(10, -2, length = 100)
ridge_mod <- glmnet(x, y, alpha = 0, lambda = grid)
```

Por defecto, la función `glmnet()` estandariza las variables predictoras para que estén en la misma escala. Si por alguna razón no se quisiera hacer, se usaría `standardize = FALSE`.

Asociado con cada valor de λ hay un vector de coeficientes estimados mediante regresión ridge almacenados en un matriz accesible utilizando `coef()`. En este caso, el tamaño de la matriz es 20×100 , donde las 20 filas corresponden a cada uno de los predictores más la ordenada en el origen y las 100 columnas a cada valor de λ . Lo esperable es que los coeficientes estimados sean más pequeños cuanto mayor sea el valor de λ . A continuación, se muestra el valor de los coeficientes cuando $\lambda = 11,498$, así como la suma de sus cuadrados, $\sum_{j=1}^p \beta_j^2$:

```
ridge_mod$lambda[50]
#> [1] 11497.57
sum(coef(ridge_mod)[-1, 50]^2)
#> [1] 40.45739
```

Por el contrario, si λ es más pequeño, 705, el valor de su suma de cuadrados es mucho mayor.

1.3. Métodos shrinkage

21

```
ridge_mod$lambda[60]
#> [1] 705.4802
sum(coef(ridge_mod)[-1, 60]^2)
#> [1] 3261.554
```

La Fig. 1.3 muestra el efecto de λ en los coeficientes del modfelo:

```
plot(ridge_mod, xvar = "lambda", label = TRUE)
```

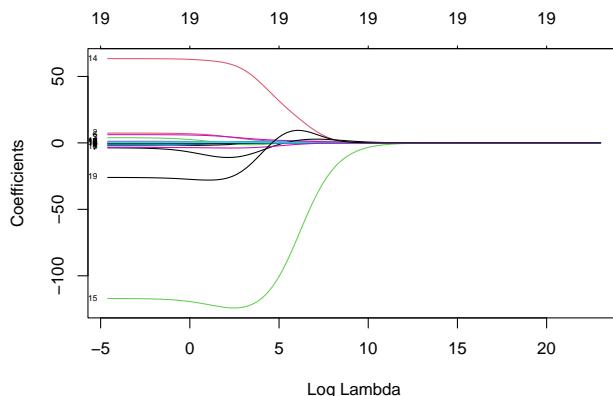


Figura 1.3: Coeficientes estimados para distintos valores del parámetro de penalización (en la escala logarítmica)

El lado izquierdo de la Fig 1.3 corresponde a valores de λ muy pequeños, y por lo tanto no existen restricciones sobre los coeficientes. Conforme aumenta el valor de λ los coeficientes se aproximan rápidamente a cero. Pero no todos se aproximan a cero de la misma manera: hay un conjunto de variables cuyo coeficiente es prácticamente cero para cualquier valor de λ , mientras que para un valor de $\log(\lambda) = 3$ parece que hay sólo 4 coeficientes distintos de 0.

La función `predict()` se puede utilizar con diferentes propósitos. Por ejemplo, se pueden obtener los coeficientes de la regresión *ridge* para un valor específico de λ , por ejemplo $\lambda = 50$:

```
predict(ridge_mod, s = 50, type = "coefficients")[1:20, ]
```

En lo que sigue, se ilustra el ajuste de una regresión ridge y se computa el *MSE* de predicción para distintos valores de λ . Primeramente, se divide el conjunto de datos en un subconjunto de entrenamiento y otro de test:

```
set.seed(1)
entreno <- sample(1:nrow(x), nrow(x) / 2)
test <- (-entreno)
y_test <- y[test]
```

A continuación se ajusta la regresión ridge a los datos del subconjunto de entrenamiento usando un valor específico de λ (por ejemplo $\lambda = 4$). Posteriormente, se evalúa su *MSE* con los datos del subconjunto de test. Para ello se usa la función `predict()`. En este caso, para obtener las predicciones para la muestra de test, se reemplaza `type = "coefficients"` por el argumento `newx`.

```
ridge_mod <- glmnet(x[entreno, ], y[entreno], alpha = 0, lambda = grid)
ridge_pred <- predict(ridge_mod, s = 4, newx = x[test, ])
mean((ridge_pred - y_test)^2)
#> [1] 142226.5
```

El *MSE* es 142,199. Si se usa un valor muy alto de λ , por ejemplo 10^{10} (esto sería equivalente a ajustar un modelo solo con la ordenada en el origen), el resultado es muy distinto:

```
ridge_pred <- predict(ridge_mod, s = 1e10, newx = x[test, ])
mean((ridge_pred - y_test)^2)
#> [1] 224669.8
```

Por lo tanto, en este caso, ajustar un modelo de regresión ridge con $\lambda = 4$ da un *MSE* mucho menor que el obtenido cuando el modelo sólo contiene la ordenada en el origen.

A continuación se compara el resultado para $\lambda = 4$ con el obtenido utilizando mínimos cuadrados ordinarios ($\lambda = 0$).³

```
ridge_pred <- predict(ridge_mod, s = 0, newx = x[test, ], exact = T,
                      x = x[entreno, ], y = y[entreno])
mean((ridge_pred - y_test)^2)
#> [1] 167018.2
```

Se observa que el *MSE* es menor cuando se usa regresión *ridge* (con $\lambda = 4$) que cuando se usan mínimos cuadrados ordinarios.

Hasta ahora se ha elegido el valor $\lambda = 4$ de forma arbitraria. En la siguiente sección se aborda la cuestión de cómo seleccionar el valor de dicho parámetro de una forma automática.

1.3.2. Selección del parámetro de penalización

En la subsección anterior se ha visto que el valor de λ tiene un gran impacto en los resultados obtenidos cuando se utiliza un modelo con penalización.

³Además se ha de añadir ‘exact = T’ en la función ‘predict()’

1.3. Métodos shrinkage

23

Una buena manera de elegir λ es usar validación cruzada (*cross-validation*). Por ejemplo, se puede usar validación cruzada con 10 grupos (*k-fold cross-validation*) :

- Se dividen los datos en k grupos, se ajusta el modelo ridge a $k - 1$ de esos grupos (para una rejilla de valores de λ) y se calcula el error de predicción para el otro grupo.
- La acción anterior se repite tomando como muestra de test cada uno de los k grupos y se suman los errores de predicción.
- Al final se dispondrá de una curva con los errores para cada valor de λ y se elegirá el que dé el mínimo error.

En la práctica, el procedimiento anterior se puede hacer con la función `cv.glmnet()`. Por defecto, esta función usa un *10-fold cross-validation*, pero el número de grupos se puede cambiar usando el argumento `nfolds`.

En el ejemplo del béisbol:

```
set.seed(1)
cv_out <- cv.glmnet(x[entreno, ], y[entreno], alpha = 0)
plot(cv_out)
```

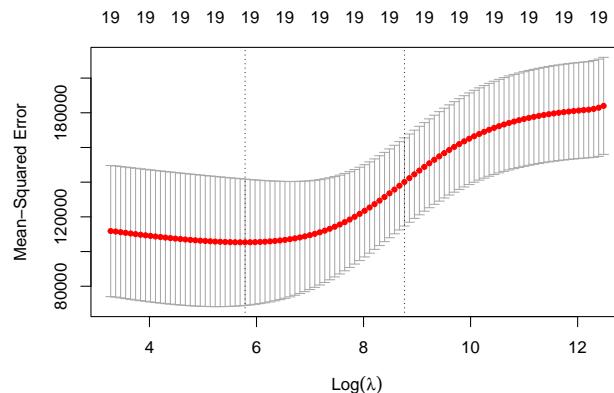


Figura 1.4: Valor del error cuadrático medio y su intervalo de confianza (calculado sobre los 10 grupos) para distintos valores del parámetro de penalización

```
mejorlam <- cv_out$lambda.min
mejorlam
#> [1] 326.0828
```

En la Fig. 1.4, los puntos rojos corresponden a la media del MSE para los 10 grupos y las barras superior e inferior corresponden a esa cantidad más/menos una desviación estándar (el ancho

será tanto menor cuanto mayor sea el número de grupos). La primera línea vertical corresponde al valor de λ que hace mínimo el MSE y la segunda es el valor que está a una distancia de una desviación típica del λ mínimo (usar este último valor podría ser una buena opción para evitar el sobre-ajuste, es decir dejar demasiadas variables en el modelo).

El valor mínimo del MSE se calcula como sigue:

```
ridge_pred <- predict(ridge_mod, s = mejorlam, newx = x[test, ])
mean((ridge_pred - y_test)^2)
#> [1] 139833.6
```

Como se puede apreciar, hay una apreciable mejora en el error de predicción que se había obtenido cuando el parámetro de penalización se había fijado en $\lambda = 4$.

1.3.3. Regresión *lasso*

Uno de los puntos débiles de la regresión *ridge* es que no hace selección de variables (los coeficientes pueden estar próximos a cero pero no ser exactamente cero). En el modelo final se incluyen todos los coeficientes y, por lo tanto, **la regresión *ridge* sólo es útil cuando la mayoría de las variables predictoras tienen un impacto significativo en la respuesta.**

La regresión *lasso* (least absolute shrinkage and selection operator, por sus siglas en inglés), introducida por Tibshirani (1996), es una alternativa a la regresión *ridge* cuyo objetivo es precisamente corregir la limitación anteriormente mencionada de la regresión *ridge*, y es útil cuando la mayoría de las variables predictoras no son relevantes en el modelo. Los coeficientes *lasso*, $\hat{\beta}^L$, minimizan la siguiente cantidad:

$$\sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^p |\beta_j| = RSS + \lambda \sum_{j=1}^p |\beta_j|.$$

Ahora los coeficientes se *contraen* hacia cero utilizando la suma de los coeficientes en valor absoluto en vez de la suma de los cuadrados de dichos coeficientes. A esta norma se le llama l_1 , $\|\beta\|_1 = \sum_{j=1}^p |\beta_j|$. El cambio que supone es sutil pero importante. En ambos casos los coeficientes se contraen hacia 0, pero en el caso de la regresión *lasso* cuando λ es suficientemente grande los coeficientes serán 0, de modo que se está haciendo una selección de variables. Por consiguiente, la regresión *lasso* anulará los coeficientes de las variables que no son importantes a la hora de explicar el comportamiento de la variable respuesta mediante un valor de λ es suficientemente grande. En este sentido el modelo de regresión *lasso* es lo que se llama un **modelo sparse** (un modelo con un número *sparse*, o escaso, de parámetros).

¿Por qué *lasso* hace que los coeficiente se contraigan exactamente hacia cero? Para entenderlo se va a ver una formulación equivalente a la de los mínimos cuadrados penalizados en el caso de la regresión *lasso*:

$$\sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 \text{ sujeto a } \sum_{j=1}^p |\beta_j| < s.$$

1.3. Métodos shrinkage

25

Dicha formulación equivalente corresponde a mínimos cuadrados con una restricción, o lo que es lo mismo, con un *presupuesto* en la norma l_1 sobre los coeficientes. Las dos formulaciones son equivalentes en el sentido de que si se tiene un *presupuesto* s , habrá un λ en la primera formulación que corresponda al presupuesto s en la segunda, y viceversa. Supóngase que se hacen mínimos cuadrados y se obtienen las estimaciones de los parámetros (coeficientes) tal que la suma de sus valores absolutos es 10, pero alguien dice que nuestro *presupuesto* es 5 (la suma de los valores absolutos de los coeficientes no puede ser mayor que esa cantidad). Entonces, hay que resolver el problema de mínimos cuadrados pero los coeficientes no pueden tomar cualquier valor, ya que se tiene una restricción sobre los mismos. Cuanto más pequeño sea el *presupuesto*, más próximos a cero serán los coeficientes. Si el *presupuesto* es 0, todos los coeficientes serán también 0. Si el presupuesto es muy alto, hay libertad para que los coeficientes tomen el valor que quieran, y se estaría en el caso de mínimos cuadrados. El *presupuesto* impone que haya un equilibrio entre el ajuste a los datos y el tamaño de los coeficientes.

La Fig. 1.5 (tomada de [James et al. \(2013\)](#)) muestra por qué el modelo de regresión lasso es *sparse*. El gráfico corresponde a un modelo de regresión con dos variables predictoras. El punto donde está el vector de coeficientes, $\hat{\beta}$, es donde se alcanza el valor mínimo de la suma los cuadrados de los residuos del modelo (RSS) y los contornos son combinaciones de valores de β_1 y β_2 que dan lugar al mismo valor de RSS , pero que ya no sería el mínimo. Las regiones de restricción son $|\beta_1| + |\beta_2| < s$ (*lasso*) y $\beta_1^2 + \beta_2^2 < s$ (*ridge*). En el caso de la regresión *ridge*, el *presupuesto* es el radio del círculo y la regresión *ridge* busca el primer lugar en el que el contorno toca a la región de restricción, pero, al ser un círculo, difícilmente uno u otro coeficiente va a ser 0. En el caso de la regresión *lasso*, la región de restricción tiene forma de diamante y, por lo tanto, tiene vértices. Como puede apreciarse, en la Fig. 1.5 el contorno toca a la región de restricción en el caso en que $\beta_1 = 0$.

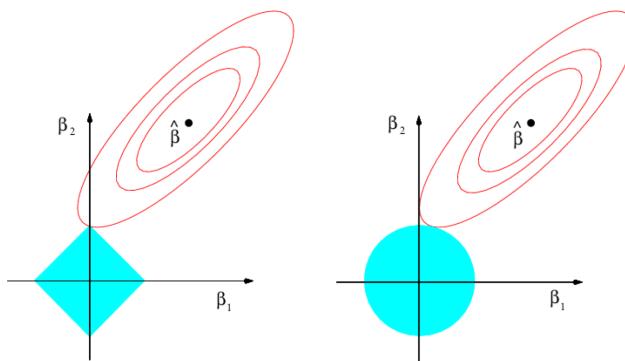


Figura 1.5: Contornos (rojo) de RSS y regiones de restricción (en azul) para la regresión lasso (izquierda) y ridge (derecha)

Se vuelve al ejemplo del béisbol para mostrar la regresión lasso; en este caso el argumento α toma valor 1 (0 en el caso de la regresión *ridge*).

```
lasso_mod <- glmnet(x[entreno, ], y[entreno], alpha = 1, lambda = grid)
plot(lasso_mod)
```

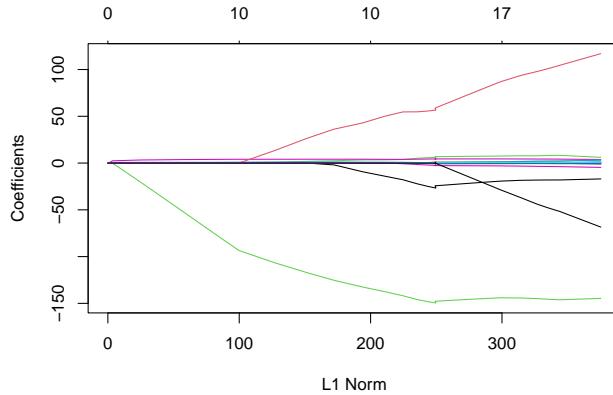


Figura 1.6: Valor de los parámetros estimados para distintos valores de la penalización (que depende del parámetros de penalización)

En la Fig. 1.6 se puede ver que, dependiendo del valor del parámetro de penalización, algunos de los coeficientes se hacen exactamente 0. Para elegir el valor de dicho parámetro y calcular el MSE resultante en el conjunto de test se procede como sigue:

```
set.seed(1)
cv_out <- cv.glmnet(x[entreno, ], y[entreno], alpha = 1)
plot(cv_out)
```

```
mejorlab <- cv_out$lambda.min
lasso.pred <- predict(lasso_mod, s = mejorlab, newx = x[test, ])
mean((lasso.pred - y_test)^2)
#> [1] 143673.6
```

Este valor es bastante más bajo que MSE en la muestra de test en el caso de mínimos cuadrados ordinarios (224,666,8) y bastante parecido al obtenido con la regresión *ridge* cuando el parámetro de penalización se elige mediante validación cruzada: 139,856,6). Sin embargo, la regresión *lasso* tiene una ventaja importante con respecto a la regresión *ridge* ya que los coeficientes estimados son *sparse*. En los resultados que se muestran a continuación, se puede observar que 10 de los 20 coeficientes estimados son 0. Por lo tanto, el modelo *lasso* con λ elegido mediante validación cruzada contiene sólo nueve variables predictoras.

1.3. Métodos shrinkage

27

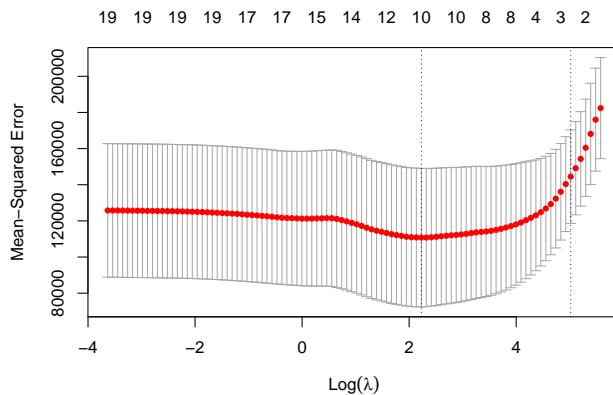


Figura 1.7: Valor del error cuadrático medio y su intervalo de confianza para distintos valores del parámetro de penalización

```
out <- glmnet(x, y, alpha = 1)
lasso_coef <- predict(out, type = "coefficients", s = mejorlab)[1:20, ]
lasso_coef[lasso_coef != 0]
#>   (Intercept)      Hits       Walks      CHmRun      CRuns
#> -3.04787656  2.02551572  2.26853781  0.01647106  0.21177390
#>      CRBI      LeagueN     DivisionW     PutOuts      Errors
#>  0.41944632 20.48456551 -116.59062083  0.23718459 -0.94739923
```

1.3.4. *Elastic net*

Uno de los problemas de la regresión *lasso* es cuando hay variables predictoras correladas entre sí, pues elegirá una de ellas (y los coeficientes de las demás los hará cero) sin un criterio objetivo. Además, supóngase que se está en una situación en la que el número de variables p es mayor que el número de observaciones n ; en este caso la regresión *lasso* elegiría como mucho n variables; mientras que la regresión *ridge* las utilizaría todas, aumentando la complejidad del modelo (esto en algunos casos puede ser lo deseable o no). *Elastic net* (Zou and Hastie, 2005) es una generalización de los métodos anteriores que combina las penalizaciones de las regresiones *ridge* y *lasso*:

$$\sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda_1 \sum_{j=1}^p \beta_j^2 + \lambda_2 \sum_{j=1}^p |\beta_j|.$$

También aparece en muchas ocasiones de esta otra forma:

$$\sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \left[\frac{1}{2}(1-\alpha) \sum_{j=1}^p \beta_j^2 + \alpha \sum_{j=1}^p |\beta_j| \right],$$

donde $\alpha \in [0, 1]$. El parámetro α es que gobierna la combinación de las dos penalizaciones, mientras que λ es el que controla la cantidad de penalización. Si $\alpha = 0$ se está en el caso de la regresión *ridge*; $\alpha = 1$ lleva a la regresión *lasso*.

La función `glmnet()` también sirve para ajustar *elastic net*, pero el parámetro α hay que elegirlo a priori. Otra opción es utilizar el paquete `caret` para hacer validación cruzada sobre α y λ simultáneamente:

```
set.seed(1)
library("caret")
cv_glmnet <- train(
  x = x[entreno, ],
  y = y[entreno],
  method = "glmnet",
  trControl = trainControl(method = "cv", number = 10),
  tuneLength = 10
)
# modelo con el MSE más pequeño
cv_glmnet$bestTune
#>   alpha    lambda
#> 9    0.1 99.12337
ggplot(cv_glmnet)
```

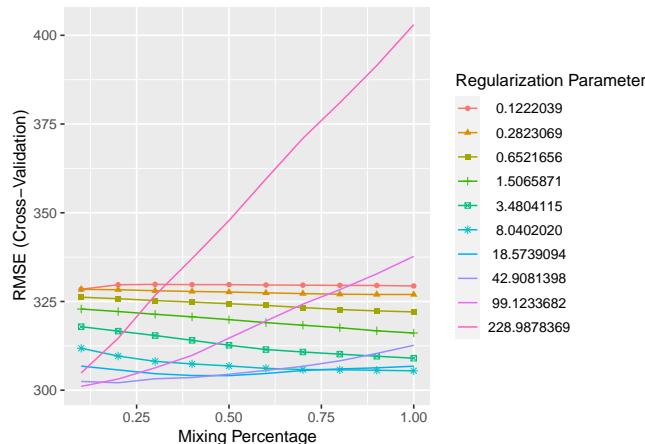


Figura 1.8: Valor de la raíz cuadrada del error cuadrático medio para distintas combinaciones de α y λ

La Fig. 1.8 muestra como la combinación de α y λ da lugar a diferentes MSE (en la figura aparece el $RMSE$, o sea, su raíz cuadrada). Cada línea corresponde a un valor de λ distinto, y en el eje x se representan los valores de α .

A continuación se calcula el valor del MSE en el conjunto de test para el modelo *elastic net* con $\alpha = 0, 1$ y $\lambda = 99, 337$, que son los valores de α y λ que hacen mínimo dicho MSE :

1.3. Métodos shrinkage

29

```
elastic_mod <- glmnet(x[entreno, ], y[entreno], alpha = cv_glmnet$bestTune$alpha)
elastic_pred <- predict(elastic_mod, newx = x[test, ], s = cv_glmnet$bestTune$lambda)
mean((elastic_pred - y_test)^2)
#> [1] 141626.1
```

Como puede comprobarse, es peor que el de la regresión *ridge* pero mejor que el de *lasso*.

Por tanto, si no se quiere hacer ningún tipo de selección debe estimar una regresión *ridge* y si se si se quiere reducir al máximo el número de variables predictoras se debe estimar una regresión *lasso* (a costa de que aumente el *MSE* en el conjunto de test). El equilibrio viene de la mano del modelo **elastic-net**, que hace selección de variables pero no aumenta el *MSE* de predicción.

Existen otros métodos de penalización que se derivan de estos, como el *group lasso*, el *sparse group-lasso*, etc. Se puede encontrar información sobre ellos en ([Hastie and Tibshirani, 2015](#)).

Resumen

En este capítulo se introducen una serie de técnicas para mejorar la predicción y la interpretabilidad de los modelos de regresión. En particular:

- Se muestra el uso de la técnica de selección del mejor subconjunto de variables en el modelo, así como los métodos *stepwise*.
- Se presentan 3 métodos tipo *shrinkage*: regresión *ridge*, *lasso* y *elastic net*, bien para la selección de variables, o para solventar problemas de multicolinealidad en el modelo.
- Se muestra cómo seleccionar el parámetro de penalización (o de combinación de penalizaciones en el caso de la regresión *elastic net*) que controla la regresión penalizada.
- Se ilustra el uso de todas las metodologías propuestas en el capítulo mediante el análisis de un caso práctico.

Bibliografía

Hastie, T. and Tibshirani, R. (2015). *Statistical Learning with Sparsity: The Lasso and Generalizations*. Monographs on Statistics & Applied Probability. Chapman and Hall/CRC.

James, G., Witten, D., Hastie, T., and Tibshirani, R. (2013). *An introduction to statistical learning*, volume 112. Springer. <https://www.statlearning.com/>.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B*, 58:267–288.

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society. Series B*, 67:301–320.

Índice alfabético

componentes principales, 12

deviance, 12

elastic net, 27

estandarización, 19

k-fold cross-validation, 23

modelo

sparse, 24

penalización shrinkage, 19

regresión lasso, 24

regresión ridge, 18

selección de variables, 11

selección stepwise, 15

backward, 15

forward, 15

Shrinkage, 12

shrinkage, 18

subconjunto

testing, 17

training, 17

validación cruzada k-grupos, 23