Mayo 2018

Cristina Gil Martínez

TÉCNICAS DE REGULARIZACIÓN Y SELECCIÓN DEL MEJOR MODELO

Apuntes personales sobre métodos de selección de predictores y mejor modelo final (en escenarios de regresión) con ejemplos en R.

CONTENIDO

INTRODUCCIÓN 1

SELECCIÓN DEL MEJOR MODELO 2

C_p de Mallow 2

Criterio de Información de Akaike (AIC) 2

Criterio de Información Bayesiano (BIC) 3

R^2 ajustado 3

Validación cruzada 4

SUBSET SELECTION 4

Best subset selection 4

Stepwise selection 5

Forward stepwise selection 5

Backward stepwise selection 6

Método híbrido 6

MÉTODOS DE REGULARIZACIÓN 7

Ridge regression 7

Ventajas frente a regresión por mínimos cuadrados y *best subset selection* 8

Lasso 9

Comparación entre *ridge regression* y *lasso* 9

Selección del parámetro de penalización λ 9

MÉTODOS DE REDUCCIÓN DIMENSIONAL 10

Análisis de Componentes Principales (PCA) 10

Regresión de Componentes Principales (PCR) 11

Mínimos Cuadrados Parciales (PLS) 12

CONSIDERACIONES CON DATOS DE ALTA DIMENSIONALIDAD 12

EJEMPLOS EN R 13

Best subset, *forward* y *backward stepwise selection* (*k-fold cross validation*) 13

Ridge regression y *Lasso* 17

PCR y PLS 22

Comparación de modelos 26

BIBLIOGRAFÍA 27

INTRODUCCIÓN

Los modelos de regresión lineal suelen ajustarse empleando el método de mínimos cuadrados, aunque métodos de ajuste alternativos podrían resultar en una mejor precisión de predicción e interpretabilidad del modelo:

- **Precisión de predicción**: dada una relación aproximadamente lineal entre la variable respuesta y los predictores, las estimaciones por mínimos cuadrados tendrán un *bias* bajo. Si además el número de observaciones es mucho mayor que el de predictores ($n >> p$), las estimaciones también tenderán a tener poca varianza, por lo que las predicciones de las observaciones de test mejorarán. Sin embargo, si n no es mucho mayor que p, el ajuste por mínimos cuadrados puede sufrir mucha variabilidad, lo que puede causar *overfitting* y predicciones futuras poco precisas. En este caso, reducir o limitar los coeficientes estimados puede suponer una reducción sustancial de la varianza a costa de un aumento poco significativo del *bias*, lo cual puede mejorar la predicción del modelo para observaciones no usadas en su entrenamiento.
- **Interpretabilidad del modelo**: puede darse el caso que alguna o muchas variables usadas en el modelo no estén asociadas con la variable respuesta. Incluir variables irrelevantes añade complejidad innecesaria al modelo, por lo que eliminar dichas variables puede mejorar la interpretabilidad de los resultados. El método de mínimos cuadrados raramente obtendremos coeficientes que sean exactamente 0, por lo que podemos aplicar métodos de selección de variables, proceso conocido como *feature selection*.

Entre las alternativas al uso de mínimos cuadrados se encuentran:

- ***Subset selection***: se basa en identificar un subconjunto de los p predictores que pensamos están relacionados con la variable respuesta. Una vez seleccionados, se ajusta el modelo con dichos predictores mediante mínimos cuadrados.
- **Regularización *o shrinkage***: se ajusta el modelo con todos los p predictores, y el valor de los coeficientes estimados es reducido (algunos pueden llegar a ser exactamente 0, con lo que son excluidos del ajuste). Esta reducción tiene el efecto de reducir la varianza.
- **Reducción dimensional**: este enfoque se basa en proyectar los p predictores en un subespacio M-dimensional, donde $M < p$, lo cual se consigue obteniendo M combinaciones lineales o proyecciones diferentes de las variables. Estas proyecciones son utilizadas como predictores para ajustar el modelo de regresión.

Los conceptos descritos anteriormente son aplicables tanto a la regresión lineal como a otro tipo de modelos.

SELECCIÓN DEL MEJOR MODELO

Existen determinadas métricas que podemos utilizar para seleccionar el mejor modelo de entre un conjunto de ellos con distintos predictores o número de ellos. Por ejemplo, el *residual sum of squares* RSS y R^2 no serían adecuados para seleccionar modelos con distinto número de predictores, ya que el *training error* tiende a decrecer conforme más variables se incluyen en el modelo. Debemos seleccionar el mejor modelo en base al *test error*, que debe ser estimado. Dos de las aproximaciones comunes son:

- Estimar el *test error* de manera **indirecta** haciendo un ajuste al *training error* que tenga en cuenta el *bias* debido al *overfitting*. Ejemplos: C_p, AIC, BIC y $R^2_{ajustado}$. (En esta sección se describen y aportan ecuaciones según un escenario de ajuste de un modelo lineal por mínimos cuadrados, pero también son aplicables a otros tipos de modelos). IMPORTANTE: no es conveniente emplear estos estadísticos si $p > n$.
- Estimar el *test error* de manera **directa**, usando validación simple o validación cruzada. (Para más información sobre estos métodos ver el capítulo Validación Cruzada y Bootstrap).

C_p de Mallow

El estadístico C_p es un estimador del test MSE de un modelo ajustado por mínimos cuadrados con d predictores, obtenido mediante la ecuación

$$C_p = \frac{1}{n}\left(\text{RSS} + 2d\hat{\sigma}^2\right)$$

donde σ^2 es la estimación de la varianza del error asociado a cada observación.

Básicamente lo que hace este estadístico es añadir una penalización de $2d\sigma^2$ al *training* RSS, dado que este tiende a subestimar el *test error*. La penalización aumenta conforme aumenta el número de predictores. Este estadístico además tiende a ser bajo para modelos con un test error bajo, por lo que, el **mejor modelo** corresponde a aquel con **menor C_p**.

Criterio de Información de Akaike (AIC)

El estadístico AIC puede aplicarse a varias clases de modelo ajustados según el criterio de máxima verosimilitud o *máximum likelihood*. La ecuación con la que se calcula el AIC es

$$\text{AIC} = \frac{1}{n\hat{\sigma}^2}\left(\text{RSS} + 2d\hat{\sigma}^2\right)$$

Para modelos ajustados por mínimos cuadrados (mínimos cuadrados es un ejemplo de *máximum likelihood*), C_p y AIC son proporcionales.

Criterio de Información Bayesiano (BIC)

Para modelos ajustados por mínimos cuadrados, el estadístico BIC se calcula con la ecuación

$$\text{BIC} = \frac{1}{n}\left(\text{RSS} + \log(n)d\hat{\sigma}^2\right)$$

donde *n* es el número de observaciones, y *d* es el número de predictores.

La fórmula del BIC reemplaza $2d\sigma^2$ usado en el cálculo del C*p* por $\log(n)d\sigma^2$. Dado que log *n* > 2 cuando *n* > 7, el estadístico BIC aumenta la penalidad en modelos con muchas variables, y como resultado, selecciona modelos de menor tamaño en comparación al C*p*.

Al igual que C*p*, valores pequeños de BIC corresponde a un test error bajo, por lo que el mejor modelo será aquel con menor BIC.

R^2 ajustado

El estadístico R^2 habitual se define como 1 – RSS/TSS, donde TSS = *total sum of squares*. Debido a que el RSS siempre disminuye conforme más variables se incluyen en el modelo, R^2 siempre aumenta. El estadístico $R^2_{ajustado}$ se calcula con la siguiente ecuación

$$\text{Adjusted } R^2 = 1 - \frac{\text{RSS}/(n-d-1)}{\text{TSS}/(n-1)}$$

donde *n* es el número de observaciones y *d* es el número de predictores.

Al contrario que con C_p, AIC y BIC para los que un valor pequeño indica un *test error* bajo, cuanto mayor es el $R^2_{ajustado}$, menor su *test error*.

Una vez las variables más importantes han sido incluidas en el modelo, la adición de variables adicionales con poca relación con la variable respuesta solo añadirá ruido, y una muy pequeña reducción en el RSS. En resumen, $R^2_{ajustado}$ penaliza la inclusión de variables innecesarias.

Para más información sobre R^2 y $R^2_{ajustado}$, ver capítulo de Regresión Lineal Múltiple.

Validación cruzada

Una alternativa más directa en relación a los métodos anteriores consiste en estimar el *test error* usando los métodos de validación cruzada (calcular el error de validación). La ventaja con la que cuenta en relación al C_p, AIC, BIC y $R^2_{ajustado}$ es que nos proporciona un estimador directo del *test error*, y cuenta con menos suposiciones acerca del verdadero modelo. Además, la validación cruzada se puede utilizar en un rango más amplio de casos de selección de modelos.

SUBSET SELECTION

Los métodos de *subset selection* incluyen: *best subset selection* y *stepwise selection*.

Best subset selection

Best subset selection consiste en ajustar un modelo por separado para cada posible combinación de los predictores, identificando finalmente el mejor de ellos en base a una determinada métrica. Más concretamente, el algoritmo sigue los siguientes pasos:

1. Se obtiene el modelo nulo M_0 (sin predictores, donde la predicción corresponderá a la media muestral para cada observación).
2. Para k = 1, 2, ... p:
   a) Se ajustan todos los modelos $\binom{p}{k}$ que contengan exactamente k predictores (usando los datos de entrenamiento).
   b) Seleccionar el mejor (menor RSS o mayor R^2) de entre los modelos anteriores (M_k).
3. Seleccionar un único modelo de entre M_0, ..., M_p usando el error de predicción por validación cruzada, C_p, AIC, BIC o $R^2_{ajustado}$.

NOTA: hay que tener cuidado si la elección del mejor modelo se basa en el RSS o R^2, ya que el primero tiende a disminuir mientras que el segundo tiende a aumentar conforme más predictores se incluyen en el

modelo (hasta llegar a un punto en el que la mejora es mínima), por lo que siempre acabaríamos eligiendo el modelo con todos los predictores. El problema es que un RSS bajo o R^2 alto indica un *training error* bajo, mientras que lo que queremos es un modelo con un *test error* bajo. Por ello el uso de validación cruzada en el paso 3.

Una **desventaja** del *best subset selection*, aun pareciendo un método simple, es la limitación computacional, ya que el número de modelos posibles crece rápidamente conforme aumenta el número de predictores (generalmente, $\mathbf{2^p}$). Por consiguiente, *best subset selection* puede llegar a ser inviable para valores de p mayores que 40 aproximadamente (según el libro ISLR). Además, cuanto mayor es el espacio de búsqueda, mayor la probabilidad de encontrar modelos que actúan bien sobre los datos de entrenamiento (*overfitting*) pero que no tienen poder predictivo para datos futuros.

A continuación, se explican alternativas más eficientes computacionalmente.

Stepwise selection

El método de *stepwise selection* o selección paso a paso supone una alternativa al *best subset selection*, e implica estudiar un número mucho menor de modelos. En este grupo se incluyen tres variantes: *forward stepwise selection*, *backward stepwise selection* y métodos híbridos.

FORWARD STEPWISE SELECTION

Mediante *forward stepwise selection* o selección paso a paso hacia delante, se comienza con el modelo nulo, sin ningún predictor para ir incluyéndolos de uno en uno hasta incluirlos todos. En cada paso la variable que aporta mayor mejora es añadida al modelo. Más concretamente, el procedimiento se define en el siguiente algoritmo:

1. Se genera el modelo nulo M_0 sin predictores.
2. Para $k = 0, ..., p - 1$:
   a) Se ajustan todos los $p - k$ modelos que aumentan M_k con un predictor adicional.
   b) Seleccionar el mejor (menor RSS o mayor R^2) de entre los modelos anteriores (M_{k+1}).
3. Seleccionar un único modelo de entre $M_0, ..., M_p$ usando el error de predicción por validación cruzada, C_p, AIC, BIC o $R^2_{ajustado}$.

En este caso, la totalidad de combinaciones de modelos que se pueden evaluar son **$1 + p(p + 1)/2$**, lo que reduce mucho el número respecto al *best subset selection* y supone una **ventaja** (por lo que es adecuado su uso cuando ***n < p***). Por contra, cuenta con la **desventaja** de que, aunque funcione bien en la práctica, el forward *stepwise selection* no garantiza encontrar el mejor modelo de entre los 2^p modelos que contienen subgrupos de los *p* predictores (con *forward stepwise selection* los predictores elegidos se van manteniendo, mientras que con *best subset selection* pueden ir cambiando).

BACKWARD STEPWISE SELECTION

Al igual que *forward stepwise selection*, *backward stepwise selection* o selección paso a paso hacia atrás, supone una alternativa más eficiente al *best subset selection*. En este caso se empieza con el modelo completo, conteniendo todos los predictores, y de manera iterativa se van eliminando uno a uno los menos útiles. El procedimiento se define con el siguiente algoritmo:

1. Se comienza con el modelo completo M_p con todos los predictores.
2. Para $k = p, p - 1, ..., 1$:
   a) Se ajustan todos los *k* modelos que contengan todos menos uno de los predictores en M_k, para un total de $k - 1$ predictores.
   b) Seleccionar el mejor (menor RSS o mayor R^2) de entre los anteriores, el cual será M_{k-1}.
3. Seleccionar un único modelo de entre $M_0, ..., M_p$ usando el error de predicción por validación cruzada, C_p, AIC, BIC o $R^2_{ajustado}$.

Al igual que con *forward stepwise selection*, se evalúan solo **$1 + p(p + 1)/2$** modelos, por lo que es adecuado para casos con un número de predictores demasiado grande para aplicar *best subset selection*. Tampoco garantiza encontrar el mejor modelo de entre los 2^p modelos que se evaluarían con *best subset selection*. A diferencia del forward stepwise selection, se requiere que el número de observaciones supere el número de predictores (***n > p***) para que el modelo completo pueda ajustarse.

Método híbrido

Best subset, *forward stepwise* y *backward stepwise selection* suelen dar modelos generalmente similares pero no idénticos. Como otra alternativa, se encuentra la versión que combina el *forward* y *backward stepwise selection* para intentar acercarse más al *best subset selection*: conforme se añaden nuevos predictores al modelo, también existe la posibilidad de eliminar aquellos que ya no mejoren el ajuste.

MÉTODOS DE REGULARIZACIÓN

Los métodos de *subset selection* comentados anteriormente implican ajustar modelos con un subconjunto de predictores. Como alternativa, podemos emplear los métodos de contracción o *shrinkage methods*: se ajusta el modelo con todos los predictores, "contrayendo" los coeficientes estimados o disminuyendo sus valores hacia 0, lo cual puede reducir de manera significativa su varianza. Las dos técnicas mejor conocidas para contraer los coeficientes de regresión son *ridge regression* y *lasso*.

Ridge regression

Si el método de mínimos cuadrados estima β_0, β_1, ..., β_p usando los valores que minimizan

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

el método de regresión contraída o *ridge regression* estima los coeficientes β^R con los valores que minimizan

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

donde **$\lambda \geq 0$** es el ***tuning parameter*** o parámetro de penalización.

Aun así, al igual que con mínimos cuadrados, *ridge regression* busca estimadores de los coeficientes que ajusten bien los datos, reduciendo el RSS. El término de la ecuación anterior que penaliza los coeficientes (**término de penalización**) es

$$\lambda \sum_{j=1}^{p} \beta_j^2$$

Se penalizan todos los coeficientes β_p a excepción del β_0 (solo queremos ajustar la asociación de los predictores con la variable respuesta).

El parámetro λ controla el impacto o el nivel de contracción sobre los coeficientes. Cuando $\lambda = 0$, el término de penalización no tiene efecto, y el resultado es equivalente al de mínimos cuadrados. Por el contrario, si **$\lambda \to \infty$**, el impacto de la penalización aumenta y los coeficientes se aproximarán más a 0 (**no llegando a**

excluirse por completo). Dependiendo del valor de λ, se producirán unos coeficientes u otros, por lo que seleccionar un valor apropiado para λ es crucial.

Los coeficientes estimados por *ridge regression* pueden verse alterados si las variables no se estandarizan antes de llevar a cabo el ajuste. Por ello, es mejor llevar a cabo esta **estandarización** para que todas estén en la misma escala (desviación estándar = 1, media = 0:

$$\tilde{x}_{ij} = \frac{x_{ij}}{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_{ij} - \overline{x}_j)^2}}$$

donde el denominador es la desviación estándar estimada para el j-ésimo predictor.

La ventaja con la que cuenta el *ridge regression* respecto al método de mínimos cuadrados reside en el equilibrio *bias*-varianza: conforme λ aumenta, la flexibilidad del ajuste por *ridge regression* disminuye, lo cual disminuye la varianza, pero aumenta el *bias*.

VENTAJAS FRENTE A REGRESIÓN POR MÍNIMOS CUADRADOS Y BEST SUBSET SELECTION

De manera general, en situaciones donde la relación entre la variable respuesta y los predictores es de tipo lineal o próximo a lineal, los estimadores mínimo cuadráticos tendrán poco *bias* pero pueden alta varianza. Ello significa que un pequeño cambio en el set de datos de entrenamiento puede hacer variar los coeficientes de manera sustancial. En particular, cuando p es casi igual de grande que *n*, los estimadores son extremadamente variables. Si $p > n$, los estimadores mínimo cuadráticos ni siquiera tienen un único valor, mientras que ***ridge regression*** **puede reducir considerablemente la varianza a expensas de un pequeño aumento en el *bias* (conforme λ aumenta)** . Por lo tanto, *ridge regression* es una mejor opción en escenarios donde $p > n$.

Ridge regression también cuenta con la ventaja computacional frente al *best subset selection*, el cual requiere evaluar 2^p modelos. Para cualquier valor de λ, *ridge regression* solo necesita ajustar un único modelo, y el ajuste se lleva a cabo bastante rápidamente.

Lasso

La inclusión en un modelo de un gran número de predictores puede suponer un desafío a la hora de su interpretación. Con *ridge regression*, la magnitud de los coeficientes se reduce, pero las variables no llegan a excluirse del modelo, y esto puede suponer una desventaja en este escenario. Aquí el método de *lasso* supone una alternativa. Los coeficientes de *lasso* minimizan

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

A diferencia del *ridge regression*, si λ es lo suficiente elevado, algunos de los coeficientes pueden llegar a reducirse hasta exactamente 0, por lo que el método actúa de alguna manera como selector de variables, y la interpretación del modelo se facilita al crear modelos más simples. De nuevo, seleccionar un valor apropiado para λ es muy importante.

Comparación entre *ridge regression* y *lasso*

Al eliminar variables del modelo, *lasso* produce modelos más simples e interpretables que *ridge regression*. Sin embargo, un caso en el que *ridge regression* podría superar a *lasso* es cuando todos los predictores están relacionados con la variable respuesta, y cuando la eliminación de variables del modelo por parte de *lasso* aumentaría el error de predicción. Pero cuando no todos los predictores se relacionan con la variable respuesta (esto es algo que no se suele conocer a *priori*), *lasso* tiende a dar mejores resultados que *ridge regression* en cuanto al *bias*, varianza y MSE.

En ambos casos, conforme λ aumenta, la varianza se reduce y el *bias* aumenta.

Utilizar la validación cruzada puede ser útil a la hora de determinar qué método funciona mejor para un set de datos determinado.

Selección del parámetro de penalización λ

La implementación de *ridge regression* y *lasso* requiere de un método para seleccionar el valor del parámetro de penalización λ. Una opción simple es utilizar validación cruzada. Se eligen un conjunto de

valores para λ y se calcula el error de validación para cada valor. se elige el valor de λ para el que el error ha sido menor, y finalmente se reajusta el modelo con todas las observaciones disponibles con el valor de λ escogido.

MÉTODOS DE REDUCCIÓN DIMENSIONAL

Mientras que los métodos anteriores mantienen las variables originales (estandarizadas), los métodos de reducción dimensional **transforman las variables** antes de ajustar el modelo, en concreto, utilizan M < *p* combinaciones lineales de los predictores. Este enfoque reduce el problema de estimar *p* + 1 coeficientes (β_0, β_1, ..., β_p) a sólo estimar M + 1 (ϑ_0, ϑ_1, ... , ϑ_M).

En situaciones donde *p* es relativamente mayor a *n*, seleccionar M << *p* puede reducir la varianza de los coeficientes de manera significativa.

La transformación de predictores (Z_1, Z_2, ..., Z_M) puede llevarse a cabo mediante dos métodos: componentes principales y mínimos cuadrados parciales.

Análisis de Componentes Principales (PCA)

El análisis de componentes principales (*principal component analysis*) o PCA es una técnica de reducción dimensional de una matriz *n* x *p*, que puede emplearse para casos como regresión o como una herramienta de aprendizaje no supervisado (ya que la variable respuesta no se utiliza para determinar la dirección de las componentes).

Se identifican las combinaciones lineales que mejor representan los predictores $\mathbf{X_1, ..., X_p}$. Sean (**$Z_1, Z_2, ..., Z_M$**) M < *p* **combinaciones lineales** de los *p* predictores originales, es decir

$$Z_m = \sum_{j=1}^{p} \phi_{jm} X_j$$

donde ***$\phi_{1m}, \phi_{2m}, ..., \phi_{pm}$*** son las constantes o ***loadings*** de los componentes principales (por ejemplo, ϕ_{11} correspondería al primer *loading* de la primera componente principal). Los *loadings* dan idea sobre qué peso tiene cada variable en cada componente.

La combinación lineal se normaliza para no inflar la varianza, por lo que

$$\sum_{j=1}^{p} \phi_{jm} X_j = 1$$

Generalmente, se podrán obtener tantas componentes principales distintas como predictores disponibles:

La **primera componente principal** (Z_1) es aquella cuya dirección refleja o contiene la mayor variabilidad en los datos (por lo que esta componente será la que mayor información contenga). Este vector define la línea lo más próxima posible a los datos y que minimiza la suma de las distancias perpendiculares entre cada dato y la línea representada por la componente.

La **segunda componente principal** (Z_2) será una combinación lineal de las variables, que recoja la segunda dirección con mayor varianza de los datos, pero que no esté correlacionada con Z_1. Esta condición es equivalente a decir que la dirección de Z_2 ha de ser **perpendicular** u **ortogonal** respecto a Z_1.

NOTA: es importante la previa **estandarización de las variables** (si no están medidas con las mismas unidades), ya que de lo contrario, los predictores con mayor varianza dominarán al resto.

Regresión de Componentes Principales (PCR)

La regresión de componentes principales se basa es ajustar un modelo de regresión utilizando las componentes obtenidas por PCA como predictores para ajustar el modelo de regresión por mínimos cuadrados. La idea clave es que con frecuencia tan solo un pequeño número de componentes principales es suficiente para explicar la mayoría de variabilidad en los datos, así como la relación con la variable respuesta.

*Si las condiciones subyacentes al PCR se cumplen, el ajuste por mínimos cuadrados sobre Z_1, ..., Z_M dará mejores resultados que usando X_1, ..., X_p, ya que toda o casi toda la información relacionada con la variable respuesta presente en los datos estará contenida en Z_1, ..., Z_M, y estimando solo M << *p* coeficientes podemos reducir el *overfitting*.

Con la inclusión de más componentes principales en el modelo de regresión, el *bias* decrece, pero la varianza aumenta. Llevar a cabo un PCR con una elección apropiada de M puede mejorar el ajuste respecto a mínimos cuadrados. La **validación cruzada** suele ser la opción elegida para escoger M.

PCR es más próximo al *ridge regression* que a *lasso*, ya que no existe selección de variables.

¿Cuándo es apto su uso?

PCR dará mejores resultados en casos donde tan solo unas pocas componentes principales sean suficientes para capturar la mayoría de varianza en los predictores y la relación con la variable respuesta.

Mínimos Cuadrados Parciales (PLS)

La regresión de componentes principales cuenta con una desventaja: al ser el PCA un método de aprendizaje no supervisado (y el PCR emplea estas componentes para la regresión), no existen garantías de que las direcciones que mejor explican la varianza de los predictores sean también las mejores para predecir la variable respuesta. **PLS supone una alternativa al PCR de aprendizaje supervisado**.

Al igual que PCR, PLS es un método de reducción dimensional que identifica un nuevo conjunto de predictores Z_1, ..., Z_M que son combinación lineal de los predictores originales para ajustar el modelo de regresión. Sin embargo, hace uso de la variable respuesta Y para su identificación (lo cual puede reducir el *bias*, pero incrementar la varianza.

NOTA: También es importante la **estandarización previa** de los predictores.

PLS identifica la dirección de la primera componente Z_1 ajustando cada ϕ_{j1} en

$$Z_m = \sum_{j=1}^{p} \phi_{jm} X_j$$

igual a los coeficientes obtenidos por regresión simple de Y sobre X_j, dando mayor peso a los predictores más fuertemente relacionados con la variable respuesta.

La dirección de los componentes no se ajusta tanto a los predictores como lo hace PCA, pero actúan mejor explicando la variable respuesta.

La **validación cruzada** también suele ser la opción elegida para escoger el número de M.

CONSIDERACIONES CON DATOS DE ALTA DIMENSIONALIDAD

La mayoría de técnicas estadísticas tradicionales de regresión y clasificación están destinadas a problemas donde $n > p$, o datos con pocas dimensiones. Por "dimensión" nos referimos al tamaño de p. Mientras que el número de predictores puede ser extremadamente alto, el número de observaciones suele estar limitado

por cuestiones de coste, disponibilidad de muestras u otras razones. Métodos como el de regresión por mínimos cuadrados no son adecuados en este escenario ($p > n$, $p \approx n$), ya que existe un alto riesgo de *overfitting* debido a la alta flexibilidad del ajuste.

Muchos de los modelos usados para ajustar modelos por mínimos cuadrados menos flexibles (como *stepwise selection*, *ridge regression*, *lasso*, PCR) son particularmente útiles para ajustar por regresión datos de altas dimensiones, ya que evitan en mayor medida el *overfitting*.

Un punto importante es que el *test error* tiende a incrementarse con la dimensionalidad (***curse of dimensionality***), a no ser que las variables adicionales estén realmente asociadas con la variable respuesta.

EJEMPLOS EN R

En este ejemplo trabajaremos con el set de datos *College* (estadísticas de universidades estadounidenses) para predecir el número de solicitudes (*Apps*) recibidas en función del resto de variables, aplicando todos los métodos vistos en este documento para compararlos y escoger el mejor modelo en base al test error y el nivel de interpretabilidad.

Best subset, forward y backward stepwise selection (k-fold cross validation)

- **`is.na()`** -> Identifica observaciones incompletas.
- **`na.omit()`** -> Elimina todas las observaciones (filas) que contienen observaciones incompletas en cualquier variable.
- **`regsubsets()` ->** Best subset selection o forward/backward si se especifica el argumento method = "forward" o method = "backward". Identifica el mejor modelo (combinación de predictores) que contiene un determinado número de predictores (por defecto muestra modelos hasta 8 variables, para cambiarlo, usar el argumento **`nvmax`**). El mejor modelo se determina mediante RSS.
- **`summary()`** -> Muestra el mejor conjunto de variables para cada tamaño de modelo. Devuelve R^2, RSS, $R^2_{ajustado}$, C_p y BIC.

Comenzaremos aplicando el método de *best subset selection* mediante la función *regsubsets( )*. El proceso sería exactamente igual para *forward* y *backward stepwise selection*, simplemente indicando en la función *regsubsets( )* el argumento *method = "forward"* o *method = "backward"*.

```
library(ISLR)
```

```
datos <- College
str(datos)
```

```
## 'data.frame':    777 obs. of  18 variables:
##  $ Private    : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Apps       : num  1660 2186 1428 417 193 ...
##  $ Accept     : num  1232 1924 1097 349 146 ...
##  $ Enroll     : num  721 512 336 137 55 158 103 489 227 172 ...
##  $ Top10perc  : num  23 16 22 60 16 38 17 37 30 21 ...
##  $ Top25perc  : num  52 29 50 89 44 62 45 68 63 44 ...
##  $ F.Undergrad: num  2885 2683 1036 510 249 ...
##  $ P.Undergrad: num  537 1227 99 63 869 ...
##  $ Outstate   : num  7440 12280 11250 12960 7560 ...
##  $ Room.Board : num  3300 6450 3750 5450 4120 ...
##  $ Books      : num  450 750 400 450 800 500 500 450 300 660 ...
##  $ Personal   : num  2200 1500 1165 875 1500 ...
##  $ PhD        : num  70 29 53 92 76 67 90 89 79 40 ...
##  $ Terminal   : num  78 30 66 97 72 73 93 100 84 41 ...
##  $ S.F.Ratio  : num  18.1 12.2 12.9 7.7 11.9 9.4 11.5 13.7 11.3 11.5 ...
##  $ perc.alumni: num  12 16 30 37 2 11 26 37 23 15 ...
##  $ Expend     : num  7041 10527 8735 19016 10922 ...
##  $ Grad.Rate  : num  60 56 54 59 15 55 63 73 80 52 ...
```

Es importante contar con todas las observaciones completas para cada variable, por lo que buscamos valores ausentes por si dichas observaciones han de ser eliminadas:

```
sum(is.na(datos))
```

```
## [1] 0
```

En este caso contamos con un set de datos completo, sin valores ausentes.

Antes de comenzar con el proceso, se dividen los datos en entrenamiento (2/3) y test (1/3). Los datos de test no participarán en la obtención del mejor modelo, sino en su evaluación final:

```
set.seed(1)
train <- sample(x = 1:nrow(datos), size = round(nrow(datos) * (2/3)))
datos.train <- datos[train, ]
datos.test <- datos[-train, ]
```

En este ejemplo emplearemos el método de *k-fold cross validation*, con 10 *folds*, distribuyendo las observaciones en *k* grupos de tamaño similar:

```
# Se asigna cada observación a un grupo
k <- 10
set.seed(11)
folds <- sample(x = 1:k, nrow(datos.train), replace = TRUE)
head(folds)
```

```
## [1]  3  1  6  1  1 10

# Distribución de las observaciones por cada grupo
table(folds)

## folds
##  1  2  3  4  5  6  7  8  9 10
## 40 64 62 47 58 51 50 50 56 40
```

Implementamos un *for loop* para llevar a cabo el *k-fold cross validation*. Para el j-ésimo *fold* o grupo, los elementos en el objeto *fold* que coincidan con j actuarán como set de validación, y el resto como set de entrenamiento. Se utilizan los datos de entrenamiento para identificar el mejor modelo para cada posible tamaño en base al RSS. A continuación, se lleva a cabo la predicción para cada modelo usando el grupo de test j, calculando el *test error* de cada grupo correspondiente y guardando el resultado en la posición correspondiente de la matriz cv_error.

NOTA: tener en cuenta que no se puede emplear la función *predict()* con un objeto *regsubsets()*, por lo que tenemos que desarrollar nuestro propio código de predicción.

```
pred_best_subset <- function (object ,newdata ,id ,...){
# extracción de la formula del modelo
form=as.formula (object$call [[2]])
# matriz modelo con la fórmula del modelo y los datos de test
mat=model.matrix (form ,newdata )
# extracción de los coeficientes del modelo
coefi =coef(object ,id=id)
# se almacena el nombre de los predictores
xvars =names (coefi )
# obtención de predicciones mediante product matricial entre los coeficientes del
modelo y el valor de los predictores en el test set
mat[,xvars ]%*% coefi
}

# Matriz donde se almacenará el error de validación
cv_error <- matrix(data = NA, nrow = 10, ncol = 17,
                   dimnames = list(NULL, c(1:17)))
# Cada fila corresponde a un fold, y cada columna a un tamaño de modelo
head(cv_error, 3)

##       1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17
## [1,] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
## [2,] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
## [3,] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA

library(leaps)

for (k in 1:10) {
train <- datos.train[folds != k, ]
best_subset <- regsubsets(Apps ~ ., data = datos.train, nvmax = 17)
# para cada mejor modelo escogido en el paso anterior...
for (i in 1:17) {
test <- datos.train[folds == k, ]
# uso de la función definida anteriormente para extraer las predicciones del model
```

```
o i almacenado en el objeto best_subset
pred <- pred_best_subset(object = best_subset, newdata = test, id = i)
# se calcula y almacena el test error (MSE) para cada modelo i
cv_error[k, i] <- mean((test$Apps - pred)^2)
}
}
```

A continuación, obtenemos la media del *test error* de los 10 grupos por cada tamaño de modelo:

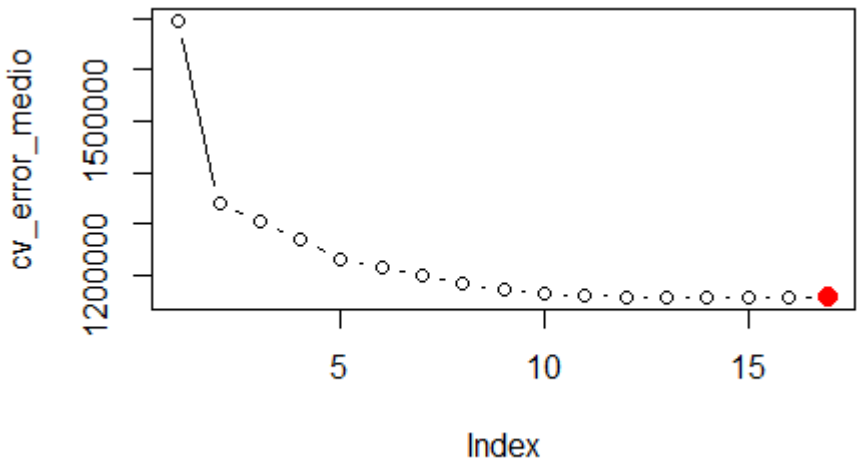
```
cv_error_medio <- apply(X = cv_error, MARGIN = 2, FUN = mean)
cv_error_medio
```

```
##       1       2       3       4       5       6       7       8       9      10

## 1696441 1339696 1303980 1270426 1231398 1214230 1199607 1183620 1172199 1163229

##      11      12      13      14      15      16      17
## 1160325 1159030 1158702 1158088 1157284 1157080 1157055
```

```
which.min(cv_error_medio)
```

```
## 17
## 17
```

```
plot(cv_error_medio, type = "b")
points(17, cv_error_medio[17], col = "red", cex = 2, pch = 20)
```



Pese a que el mínimo error se consigue con un modelo con 17 predictores, a partir de 7 predictores aproximadamente, la reducción del error es mínima, por lo que podríamos optar por este último tamaño.

```
test.MSE.subset <- cv_error_medio[7]
test.MSE.subset
```

```
##       7
## 1199607
```

Como último paso, se identifican los coeficientes de los predictores escogidos en el mejor modelo, esta vez con el set de datos completo (para obtener unos coeficientes más precisos), y se calcula su correspondiente test error.

NOTA: No se lleva a cabo el *best subset selection* con los 7 predictores escogidos con los datos de entrenamiento, porque el mejor modelo con 7 variables usando todos los datos puede no ser el mismo, es decir, puede que no coincidan las 7 variables.

```
modelo.bestsubset <- regsubsets(Apps ~ ., data = datos.train, nvmax = 17)

coef(modelo.bestsubset, 7)
```

```
##   (Intercept)        Accept        Enroll     Top10perc     Top25perc
## -455.93993540    1.64578763   -0.64049895   51.26531303  -15.13842211
##      Outstate    Room.Board        Expend
##   -0.12094977    0.16546142    0.06300329
```

Ridge regression y Lasso

- **is.na()** -> Identifica observaciones incompletas.
- **na.omit()** -> Elimina todas las observaciones (filas) que contienen observaciones incompletas en cualquier variable.
- **model.matrix()** -> Crea una matriz modelo a partir de los datos. Automáticamente transforma las variables cualitativas en variables dummy.
- **glmnet()** -> Ajuste de modelos, entre ellos, ridge regression (Alpha = 0) y lasso (alpha = 1). Necesita una matriz x y un vector y para la variable respuesta. Solo trabaja con inputs cuantitativos. Por defecto, estandariza las variables (se puede cambiar con standardize = FALSE).
- **cv.glmnet()** -> K-fold cross validation (10 folds por defecto) para glmnet().
- **predict()** -> Predicciones a partir de los resultados de funciones de ajuste de modelos.
- **coef()** -> Extracción de coeficientes.

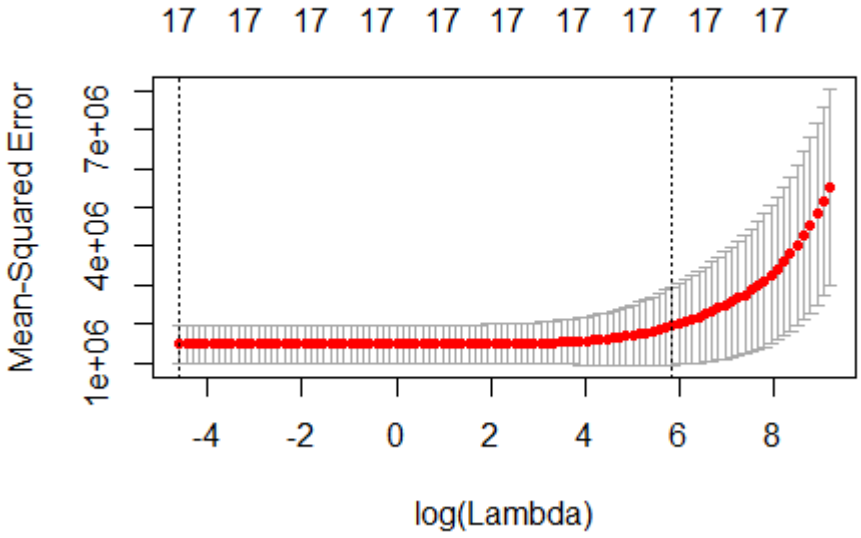
La función a utilizar en ambos casos es la misma, *glmnet()*, con la diferencia de que para aplicar *ridge regression* ha de usarse el argumento *alpha = 0*, y *alpha = 1* si se quiere aplicar *lasso*.

Antes de proceder con estos métodos, es importante asegurar la ausencia de valores ausentes (NA). En este caso vamos a continuar con el mismo set de datos, con lo que no es necesario volver a hacer esta comprobación.

El valor de lambda λ será elegido mediante validación cruzada, en lugar de elegirlo arbitrariamente, y *ridge regression/lasso* se llevarán a cabo sobre los datos de entrenamiento. Para crear la matriz de valores

numéricos que requiere la función *glmnet()*, aplicamos la función *model.matrix()* a los datos de entrenamiento y validación.

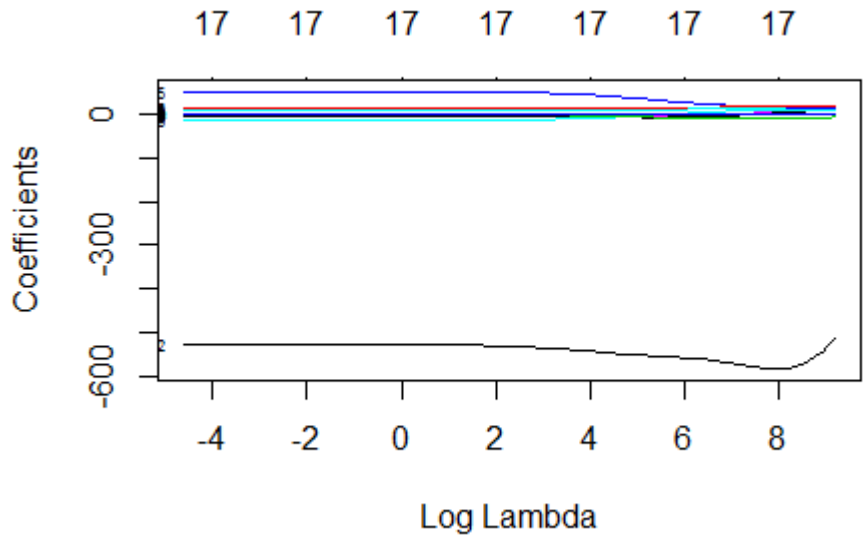
```
library(glmnet)

# Conversión a matriz modelo de los datos de train y test
datos.train.mat <- model.matrix(Apps ~ ., data = datos.train)
datos.test.mat <- model.matrix(Apps ~ ., data = datos.test)

# Conjunto de valores de lambda
lambda = 10 ^ seq(from = 4, to = -2, length = 100)
# 10-fold cross validation para obtener el mejor lambda
set.seed(12)
cv.ridge <- cv.glmnet(x = datos.train.mat, y = datos.train$Apps, alpha = 0,
                      lambda = lambda, thresh = 1e-12, type.measure="mse")

plot(cv.ridge)
```




```
names(cv.ridge)

##  [1] "lambda"     "cvm"        "cvsd"       "cvup"       "cvlo"
##  [6] "nzero"      "name"       "glmnet.fit" "lambda.min" "lambda.1se"

# Mejor lambda (menor error de validación)
cv.ridge$lambda.min

## [1] 0.01



# Modelo RIDGE con los datos de entrenamiento. Se asocia un vector de coeficientes
 para cada valor de lambda
modelo.ridge.train <- glmnet(x = datos.train.mat, y = datos.train$Apps,
                             alpha = 0, lambda = lambda, thresh = 1e-12)
# 18 variables + intercept, y 100 vectores de coeficientes (uno para cada valor de
 lambda
dim(coef(modelo.ridge.train))
```

```
## [1]  19 100
```

```
plot(modelo.ridge.train, xvar = "lambda", label = TRUE)
```



El valor de λ con el que se obtiene el menor error de validación es 0,01. Con los datos de test calculamos el test error para el modelo con este valor de penalización:

```
# Predicciones del modelo con los datos de test y el mejor lambda
pred.modelo.ridge <- predict(modelo.ridge.train, s = 0.01, newx = datos.test.mat)
# Test error (MSE)
test.MSE.ridge <- mean((pred.modelo.ridge - datos.test$Apps)^2)
test.MSE.ridge
```

```
## [1] 925308.1
```

Finalmente, reajustamos el modelo *ridge regression* con todos los datos, usando el valor de λ escogido por validación cruzada, y obtenemos los estimadores de los coeficientes:

```
# Se excluye la primera columna con los nombres de las universidades
modelo.ridge <- glmnet(x = model.matrix(Apps ~ ., data = datos)[,-1],
                       y = datos$Apps, alpha = 0)
# Coeficientes del modelo
predict(modelo.ridge, type = "coefficients", s = 0.01)
```

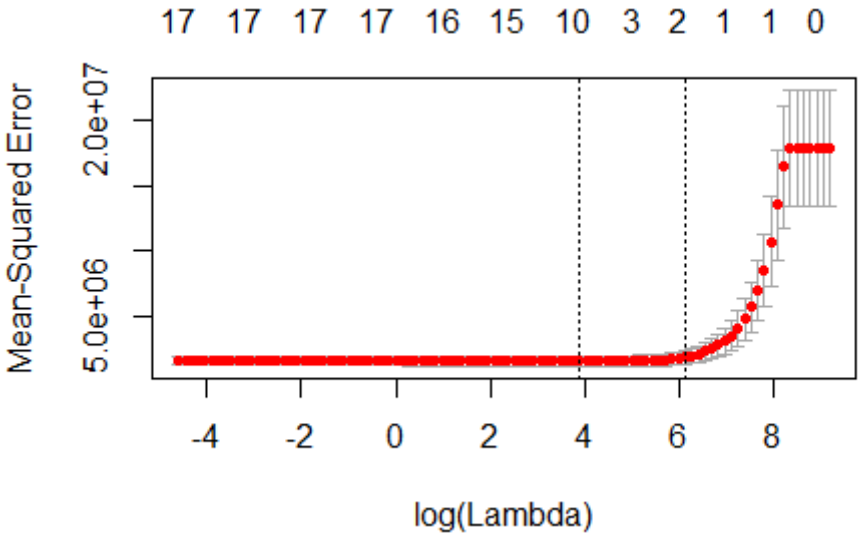
```
## 18 x 1 sparse Matrix of class "dgCMatrix"
##                          1
## (Intercept) -1.468326e+03
## PrivateYes  -5.278781e+02
## Accept       1.004588e+00
## Enroll       4.313442e-01
## Top10perc    2.580619e+01
## Top25perc    5.501092e-01
## F.Undergrad  7.258520e-02
```

```
## P.Undergrad  2.420595e-02
## Outstate    -2.407454e-02
## Room.Board   1.987732e-01
## Books        1.285477e-01
## Personal    -8.146131e-03
## PhD         -4.028284e+00
## Terminal    -4.811071e+00
## S.F.Ratio    1.302180e+01
## perc.alumni -8.544783e+00
## Expend       7.589013e-02
## Grad.Rate    1.126699e+01
```

Como podemos observar, todos los predictores han sido incluidos en el modelo, ya que *ridge regression* no reduce ningún valor de coeficiente a 0, por lo que no selecciona predictores.

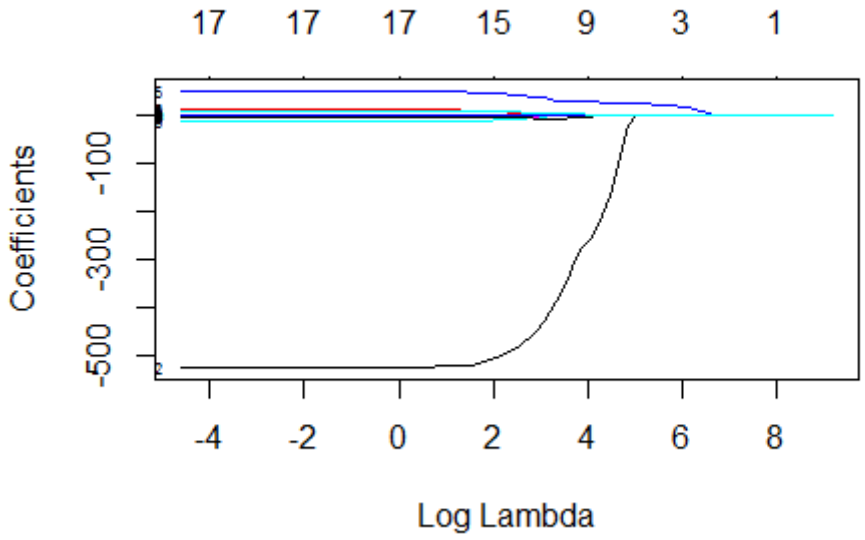
Veamos que ocurre aplicando *lasso*:

```
set.seed(11)

cv.lasso <- cv.glmnet(x = datos.train.mat, y = datos.train$Apps, alpha = 1,
                      lambda = lambda, thresh = 1e-12, type.measure="mse")

plot(cv.lasso)
```




```
mejor.lambda <- cv.lasso$lambda.min
mejor.lambda

## [1] 32.74549

# Modelo LASSO con los datos de entrenamiento. Se asocia un vector de coeficientes
 para cada valor de lambda
```

```
modelo.lasso.train <- glmnet(x = datos.train.mat, y = datos.train$Apps,
                             alpha = 1, lambda = lambda, thresh = 1e-12)

plot(modelo.lasso.train, xvar = "lambda", label = TRUE)
```




```
# Predicciones del modelo con los datos de test y el mejor lambda
pred.modelo.lasso <- predict(modelo.lasso.train, s = 49.77024,
                             newx = datos.test.mat)
# Test error (MSE)
test.MSE.lasso <- mean((pred.modelo.lasso - datos.test$Apps)^2)
test.MSE.lasso

## [1] 955985.4
```

Finalmente, reajustamos el modelo *lasso* con todos los datos, usando el valor de λ escogido por validación cruzada, y obtenemos los estimadores de los coeficientes:

```
modelo.lasso <- glmnet(x = model.matrix(Apps ~ ., data = datos)[,-1],
                       y = datos$Apps, alpha = 1)
# Coeficientes del modelo
predict(modelo.lasso, type = "coefficients", s = 49.77024)

## 18 x 1 sparse Matrix of class "dgCMatrix"
##                         1
## (Intercept) -7.405034e+02
## PrivateYes  -3.210426e+02
## Accept       1.392629e+00
## Enroll       .
## Top10perc    2.701180e+01
## Top25perc    .
## F.Undergrad  .
## P.Undergrad  .
## Outstate    -2.346016e-02
```

```
## Room.Board   6.716885e-02
## Books        .
## Personal     .
## PhD         -1.721920e+00
## Terminal    -1.746461e+00
## S.F.Ratio    .
## perc.alumni -8.300449e-04
## Expend       5.576760e-02
## Grad.Rate    1.490820e+00
```

En este caso, *lasso* ha descartado 7 predictores del modelo. Sin embargo, el test error (955985) supera al modelo generado por *ridge regression* (925308).

PCR y PLS

- **is.na()** -> Identifica observaciones incompletas.
- **na.omit()** -> Elimina todas las observaciones (filas) que contienen observaciones incompletas en cualquier variable.
- **pcr()** -> Regresión de componentes principales. Permite uso de cross validation dentro de la misma fórmula.
- **plsr()** -> Partial least squares
- **validationplot()** -> Función para trazar estadísticas de validación, como RMSEP o R ^ 2, en función del número de componentes.

(Asegurarse de que los datos con los que vamos a trabajar con PCR o PLS no contienen observaciones incompletas)

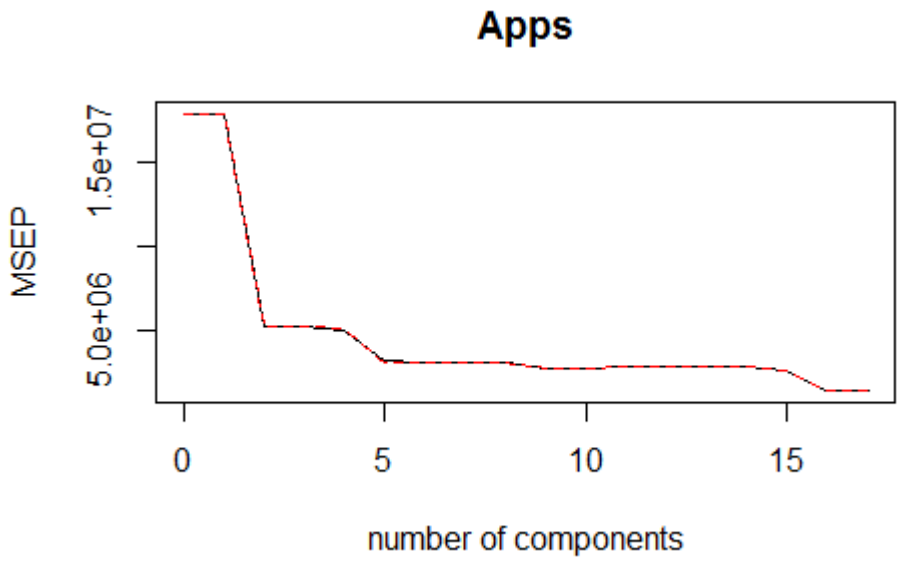
Al ajustar un modelo PCR es importante **estandarizar los predictores**. Aplicaremos además *10-fold cross validation* para calcular el error asociado a cada posible valor de M (número de componentes principales), y escoger el mejor.

```
library(pls)

# Modelo PCR
modelo.pcr.train <- pcr(Apps ~ ., data = datos.train, scale = TRUE,
                        validation = "CV")
# Resultado del ajuste del modelo
summary(modelo.pcr.train)

## Data:    X dimension: 518 17
##  Y dimension: 518 1
## Fit method: svdpc
## Number of components considered: 17
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
```

```
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV            4223     4198     2261     2245     2199     1759     1756
## adjCV         4223     4201     2259     2242     2205     1748     1751
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV        1757     1762     1666      1667      1674      1679      1685
## adjCV     1752     1757     1661      1662      1669      1674      1680
##        14 comps  15 comps  16 comps  17 comps
## CV         1679      1630      1285      1252
## adjCV      1677      1605      1273      1242
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X      30.930    57.85    64.82    70.64    76.17    81.10    84.63    87.99
## Apps    2.145    71.92    72.40    74.19    83.98    84.09    84.17    84.17
##       9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X       90.77     93.06     95.10     96.79     97.93     98.74     99.38
## Apps    85.79     85.87     85.88     85.88     85.90     86.08     91.01
##       16 comps  17 comps
## X        99.85    100.00
## Apps     93.30     93.55
```

En el *summary* del modelo se muestra el error de validación (CV y adjCV, que corresponden al RMSEP o *root mean squared error*) correspondiente al uso de distinto número de componentes. También se muestra el porcentaje de varianza explicada en los predictores y la variable respuesta.

```
validationplot(modelo.pcr.train, val.type = "MSEP")
```



Como podemos observar, el error de validación disminuye con el número de componentes, aunque a partir de 5 componentes principales, la disminución es poco significativa, por lo que sería razonable utilizar este número. Utilizar todas las componentes equivaldría a no reducir la dimensionalidad.

El test error del modelo con 5 componentes sería:

```
# Test MSE
pred.modelo.pcr <- predict(modelo.pcr.train, datos.test, ncomp = 5)
test.MSE.pcr <- mean((pred.modelo.pcr - datos.test$Apps)^2)
test.MSE.pcr

## [1] 1778741
```

Hemos obtenido un test error mayor que con *best subset selection*, *ridge regression* o *lasso*. Ajustando un modelo con PCR no obtenemos los coeficientes. Tampoco se lleva a cabo selección de predictores.

Por último, ajustamos el modelo con todos los datos, usando M = 5:

```
modelo.pcr <- pcr(Apps ~ ., data = datos, scale = TRUE, ncomp = 5)
summary(modelo.pcr)

## Data:    X dimension: 777 17
##  Y dimension: 777 1
## Fit method: svdpc
## Number of components considered: 5
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps
## X      31.670    57.30    64.30    69.90    75.39
## Apps    2.316    73.06    73.07    82.08    84.08
```

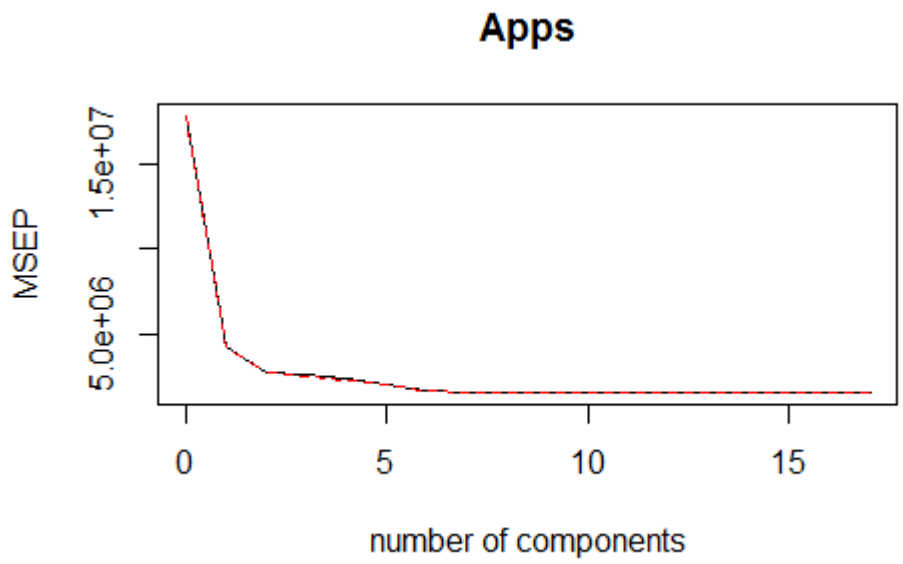
En el modelo final, las cinco componentes principales son capaces de explicar el 84,08% de la varianza en la variable respuesta *Apps*, mientras que capturan el 75,39% de la varianza en los predictores.

Por último, ajustamos el modelo mediante PLS. La sintaxis es la misma que con el uso de *pcr()*:

```
set.seed(1)
# Modelo PLS
modelo.pls.train <- plsr(Apps ~ ., data = datos.train, scale = TRUE,
                         validation = "CV")
# Resumen del ajuste del modelo
summary(modelo.pls.train)

## Data:    X dimension: 518 17
##  Y dimension: 518 1
## Fit method: kernelpls
## Number of components considered: 17
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV            4223     2063     1664     1606     1537     1423     1288
## adjCV         4223     2060     1656     1600     1519     1406     1275
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV        1271     1259     1258      1262      1262      1261      1261
## adjCV     1260     1248     1247      1251      1251      1250      1250
```

```
##        14 comps  15 comps  16 comps  17 comps
## CV         1260      1260      1260      1260
## adjCV      1249      1249      1249      1249
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X       26.92    36.26    63.09    65.86    70.29    73.79    78.38    80.76
## Apps    77.16    86.34    87.72    91.18    92.67    93.37    93.41    93.47
##       9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X       83.65     86.95     89.54     91.09     92.23     94.41     96.77
## Apps    93.51     93.52     93.54     93.55     93.55     93.55     93.55
##       16 comps  17 comps
## X        98.31    100.00
## Apps     93.55     93.55
```

```
validationplot(modelo.pls.train, val.type = "MSEP")
```



Según la evolución del MSEP, la mayor reducción se produce hasta las 2 o 3 primeras componentes.

El *test error* del modelo usando las 3 primeras componentes principales sería:

```
pred.modelo.pls <- predict(modelo.pls.train, datos.test, ncomp = 3)
test.MSE.pls <- mean((pred.modelo.pls - datos.test$Apps)^2)
test.MSE.pls
```

```
## [1] 1237333
```

El test error disminuye respecto al ajuste del modelo por PCR.

Por último, ajustamos el modelo con todos los datos, usando M = 3:

```
modelo.pls <- plsr(Apps ~ ., data = datos, scale = TRUE, ncomp = 3)
summary(modelo.pls)
```

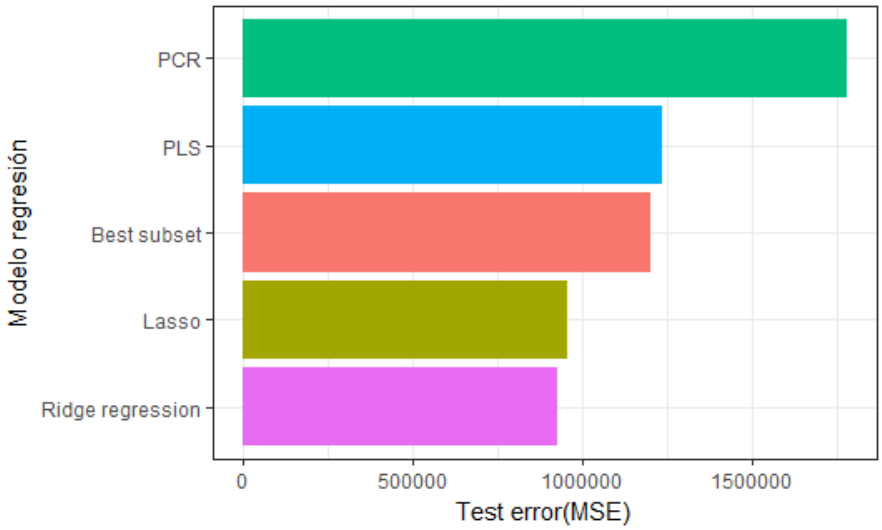
```
## Data:    X dimension: 777 17
##  Y dimension: 777 1
## Fit method: kernelpls
## Number of components considered: 3
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps
## X       25.76    40.33    62.59
## Apps    78.01    85.14    87.67
```

El porcentaje de la varianza explicada en la variable respuesta por solo las 3 primeras componentes es del 87,67%, algo mayor que lo explicado por las 5 primeras componentes en el modelo ajustado por PCR. Este modelo ha conseguido explicar mayor varianza (ya que PLS intenta maximizar la varianza explicada, no solo de los predictores, sino de la variable respuesta) con menos componentes, teniendo además un test error menor.

Comparación de modelos

Una vez se han ajustado distintos tipos de modelos para el mismo set de datos, podemos compararlos para evaluar el que mejor precisión de predicción consigue, teniendo en cuenta el nivel de interpretabilidad si la diferencia entre ellos no es muy grande en cuanto al error de predicción. En este caso, podemos comparar la métrica calculada para cada uno, test MSE:

```
modelo <- c("Best subset", "Ridge regression", "Lasso", "PCR", "PLS")
test.MSE <- c(test.MSE.subset, test.MSE.ridge, test.MSE.lasso, test.MSE.pcr, test.
MSE.pls)
comparacion <- data.frame(modelo, test.MSE)
comparacion

##             modelo  test.MSE
## 1      Best subset 1199607.1
## 2 Ridge regression  925308.1
## 3            Lasso  955985.4
## 4              PCR 1778741.5
## 5              PLS 1237332.9

library(ggplot2)
ggplot(data = comparacion, aes(x = reorder(x = modelo, X = test.MSE),
                               y = test.MSE)) +
geom_bar(stat = "identity", aes(fill = modelo)) +
labs(x = "Modelo regresión", y = "Test error(MSE)") +
theme_bw() +
coord_flip() +
theme(legend.position = "none")
```



El modelo que ha conseguido en este estudio un menor error de predicción es el ajustado mediante *ridge regression*.

BIBLIOGRAFÍA

An Introduction to Statistical Learning: with Applications in R (Springer Texts in Statistics)