Estymacja ściągająca –Shrinkage methods- regularyzacja

Klasyczna estymacja wektora $\boldsymbol{\beta}$ w modelu $\mathbf{Y}=\mathbf{X}\boldsymbol{\beta}+\boldsymbol{\varepsilon}$ polega na minimalizacji $RSS=\|\mathbf{Y}-\mathbf{X}\boldsymbol{\beta}\|^2$ gdzie $E(\boldsymbol{\varepsilon})=\mathbf{0}$ i $E(\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^{\mathrm{T}})=\sigma^2\mathbf{I}$. Trudności w estymacji parametrów klasyczną MNK wynikają czasami z odległości pomiędzy estymatorem $\hat{\boldsymbol{\beta}}=(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{Y}$ i parametrem $\boldsymbol{\beta}$. Rzeczywiście

$\hat{\boldsymbol{\beta}}=(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{Y}=(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}(\mathbf{X}\boldsymbol{\beta}+\boldsymbol{\varepsilon})=\boldsymbol{\beta}+(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\boldsymbol{\varepsilon}$. Więc . $\hat{\boldsymbol{\beta}}-\boldsymbol{\beta}=(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\boldsymbol{\varepsilon}$.

Stąd $\left\|\hat{\boldsymbol{\beta}}-\boldsymbol{\beta}\right\|^2=\boldsymbol{\varepsilon}^{\mathrm{T}}\mathbf{X}(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\boldsymbol{\varepsilon}$ i

$$E\left\|\hat{\boldsymbol{\beta}}-\boldsymbol{\beta}\right\|^2=E(\boldsymbol{\varepsilon}^{\mathrm{T}}\mathbf{X}(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\boldsymbol{\varepsilon})=E\left(\mathrm{tr}(\boldsymbol{\varepsilon}^{\mathrm{T}}\mathbf{X}(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\boldsymbol{\varepsilon})\right)=$$
$$=E\left(\mathrm{tr}(\mathbf{X}(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^{\mathrm{T}})\right)=\mathrm{tr}(\mathbf{X}(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}E(\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^{\mathrm{T}}))=$$
$$=\mathrm{tr}(\mathbf{X}(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\sigma^2\mathbf{I})=\sigma^2\mathrm{tr}\left((\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)=\sigma^2\mathrm{tr}(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}$$

$$E\left\|\hat{\boldsymbol{\beta}}\right\|^2=E(\hat{\boldsymbol{\beta}}^{\mathrm{T}}\hat{\boldsymbol{\beta}})=\boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{\beta}+\sigma^2\mathrm{tr}(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}=\|\boldsymbol{\beta}\|^2+\sigma^2\mathrm{tr}(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}$$

W terminach wartości własnych $\lambda_1\geq\lambda_2\geq\cdots\geq\lambda_p$ macierzy $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ możemy napisać

$$E\left\|\hat{\boldsymbol{\beta}}-\boldsymbol{\beta}\right\|^2=\sigma^2\mathrm{tr}(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}=\sigma^2\sum_{i=1}^{p}\tfrac{1}{\lambda_i}>\frac{\sigma^2}{\lambda_p},\ \text{bo}\ \mathrm{tr}(\mathbf{X}^{\mathrm{T}}\mathbf{X})=\sum_{i=1}^{p}\lambda_i\ \text{i}\ \mathrm{tr}(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}=\sum_{i=1}^{p}\lambda_i^{-1}\cdot$$

Widać, że w przypadku współliniowości zmiennych objaśniających mamy do czynienia ze złym uwarunkowaniem macierzy $\mathbf{X}^{\mathrm{T}}\mathbf{X}$. Bliska zeru najmniejsza wartość własna skutkuje dużą odległością pomiędzy estymatorem a prawdziwą wartością parametru.

Procedura regresji grzbietowej proponuje użyć estymatora $\hat{\boldsymbol{\beta}}^{*}=(\mathbf{X}^{\mathrm{T}}\mathbf{X}+\lambda\mathbf{I})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{Y}$

Regresja grzbietowa estymuje parametry modelu minimalizując zmodyfikowane kryterium

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

gdzie λ jest parametrem (*tuning parameter*) dobieranym zewnętrznie. Składnik $\lambda\sum_{j=1}^{p}\beta_j^2$ w powyższym kryterium zwany (*shrinkage penalty*) jest karą ściągającą parametry w kierunku 0. W odróżnieniu od klasycznej MNK regresja grzbietowa produkuje parametryczną rodzinę estymatorów $\hat{\beta}_\lambda^R$ dla każdego $\lambda\geq 0$. Są różne propozycje wyboru parametru λ. W pakiecie MASS funkcja lm.ridge() realizuje regresję grzbietową. Biblioteka glmnet zawiera zestaw funkcji realizujących m.in. regresję grzbietową i lassso łącznie z pomocniczymi funkcjami wyznaczającymi optymalny parametr lambda metodą crossvalidacji funkcja cv.glmnet()

Regresja LASSO (*Least Absolute Shrinkage and Selection Operator*) estymuje parametry modelu minimalizując kryterium

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

Inne sformułowanie regresji ridge i lasso

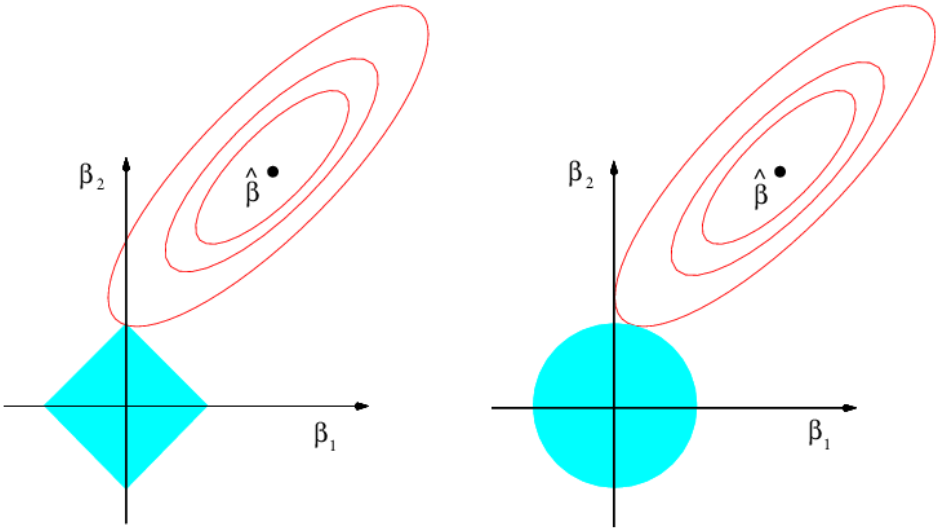
$$\min_{\beta}\sum_{i=1}^{n}(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij})^2 \text{ pod warunkiem } \sum_{j=1}^{p}\beta_j^2\le s \text{ ridge}$$

$$\min_{\beta}\sum_{i=1}^{n}(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij})^2 \text{ pod warunkiem } \sum_{j=1}^{p}|\beta_j|\le s \text{ lasso}$$

Powyższe sformułowanie pokazuje bliski związek pomiędzy tymi metodami a wyborem najlepszego podzbioru

$$\min_{\beta}\sum_{i=1}^{n}(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij})^2 \text{ pod warunkiem } \sum_{j=1}^{p}I(\beta_j\neq 0)\le s\ .$$

(czyli model może zawierać co najwyżej s predyktorów.)



Prosty przypadek specjalny dla regresji grzbietowej i lasso

Aby uzyskać lepszą intuicję dotyczącą zachowania regresji grzbietowej i lasso, rozważmy prosty przypadek specjalny, w którym $n = p$ i $\mathbf{X}$ to macierz diagonalna z jedynkami na przekątnej i zerami we wszystkich elementach nie diagonalnych. Aby jeszcze bardziej uprościć problem, załóżmy również, że wykonujemy regresję bez wyrazu wolnego. Przy tych założeniach zwykły problem najmniejszych kwadratów upraszcza się do znalezienia $\beta_1, \ldots, \beta_p$ które minimalizują

$$\sum_{j=1}^{p}(y_j-\beta_j)^2 .$$

Oczywiście MNK prowadzi do rozwiązania $\beta_j = y_j$.

Regresja grzbietowa polega na znalezieniu $\beta_1, \ldots, \beta_p$, które minimalizują

$$\sum_{j=1}^{p}(y_j-\beta_j)^2 + \lambda\sum_{j=1}^{p}\beta_j^2 \quad , \ \lambda > 0$$

natomiast lasso polega poszukiwaniu współczynników $\beta_1, \ldots, \beta_p$, które minimalizują

$$\sum_{j=1}^{p}(y_j-\beta_j)^2 + \lambda\sum_{j=1}^{p}|\beta_j| \ , \ \lambda > 0 .$$

Z uwagi na postać funkcji (separowalność zmiennych), którą należy minimalizować możemy ograniczyć się do jednej zmiennej. I tak dla regresji grzbietowej mamy problem : Znaleźć β , które minimalizuje

$$\varphi(\beta) = (y-\beta)^2 + \lambda\beta^2 .$$

Oczywiście $\hat{\beta}_{ridge} = \dfrac{y}{1+\lambda}$.

Natomiast dla regresji lasso funkcja do minimalizacji ma postać

$$\varphi(\beta) = (y-\beta)^2 + \lambda|\beta| .$$

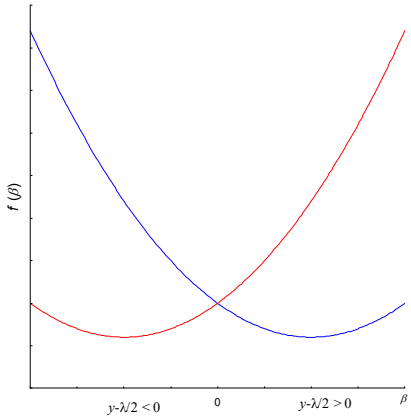
Przekształcimy ją do wygodniejszej postaci

$$\varphi(\beta) = y^2 - 2y\beta + \beta^2 + \lambda\,\mathrm{sign}(\beta)\cdot\beta = \beta^2 + (\lambda\,\mathrm{sign}(\beta) - 2y)\beta + y^2 =$$

$$\left(\beta - (y - \tfrac{\lambda}{2}\mathrm{sign}(\beta))\right)^2 + y\lambda\,\mathrm{sign}(\beta) - \tfrac{\lambda^2}{4} =$$

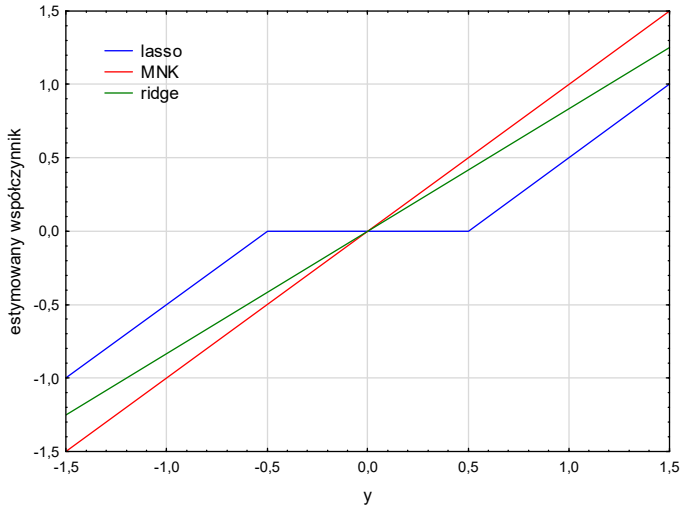
$$= \begin{cases} \left(\beta-(y-\frac{\lambda}{2})\right)^2 + y\lambda - \frac{\lambda^2}{4}, \text{dla} \ \ \beta \ge 0 \\ \left(\beta-(y+\frac{\lambda}{2})\right)^2 - y\lambda - \frac{\lambda^2}{4}, \text{dla} \ \ \beta < 0 \end{cases} = \begin{cases} \left(\beta-(y-\frac{\lambda}{2})\right)^2 + y^2 - (y-\frac{\lambda}{2})^2, \text{dla} \ \ \beta \ge 0 \\ \left(\beta-(y+\frac{\lambda}{2})\right)^2 + y^2 - (y+\frac{\lambda}{2})^2, \text{dla} \ \ \beta < 0 \end{cases}$$

Rozważmy przypadek $\beta \ge 0$.



Widać że jeśli $y - \frac{\lambda}{2} > 0$, to $\hat{\beta}_{lasso} = y - \frac{\lambda}{2}$, $y - \frac{\lambda}{2} < 0$, to $\hat{\beta}_{lasso} = 0.$

Podobnie dla $\beta \le 0$ jeśli $y+\frac{\lambda}{2}<0$, to $\hat{\beta}_{lasso}=y+\frac{\lambda}{2}$, $y+\frac{\lambda}{2}>0$, to $\hat{\beta}_{lasso}=0.$



Widzimy, że regresja grzbietowa i lasso powodują dwa bardzo różne rodzaje ściągania. W regresji grzbietowej każde oszacowanie współczynnika najmniejszych kwadratów jest zmniejszane o tę samą proporcję. W przeciwieństwie do tego lasso zmniejsza każdy współczynnik najmniejszych kwadratów w kierunku zera o stałą wartość λ/2; współczynniki najmniejszych kwadratów, które są mniejsze niż λ/2 w wartości bezwzględnej, są całkowicie ściągane do zera. Ten rodzaj ściągania wykonywany przez lasso nazywany jest *miękkim progowaniem* (*soft tresholding*). Fakt, że niektóre współczynniki lasso są całkowicie ściągnięte do zera wyjaśnia, dlaczego lasso dokonuje selekcji zmiennych

Bayesowska interpretacja regresji grzbietowej i lasso

Regresja grzbietowa

Przyjmijmy warunkowy rozkład obserwacji $Y \mid \beta \sim N(\beta,\sigma^2)$ i rozkład a priori parametru $\beta \sim N(0,\frac{\sigma^2}{\lambda})$. Wobec tego $p(y \mid \beta)=\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(y-\beta)^2}{2\sigma^2}}$ i $h(\beta)=\frac{\sqrt{\lambda}}{\sqrt{2\pi}\sigma}e^{-\frac{\lambda\beta^2}{2\sigma^2}}$. Łączny rozkład wektora (y,β) jest postaci

$$g(y,\beta)=p(y \mid \beta)h(\beta)=\frac{\sqrt{\lambda}}{2\pi\sigma^2}e^{-\frac{1}{2\sigma^2}[(y-\beta)^2+\beta^2\lambda]}.$$

Stąd

$$k(y)=\int_R g(y,\beta)d\beta=p(y \mid \beta)h(\beta)=\frac{\sqrt{\lambda}}{2\pi\sigma^2}\int_R e^{-\frac{1}{2\sigma^2}[(1+\lambda)\beta^2-2\beta y+y^2]}d\beta=$$

$$\frac{\sqrt{\lambda}}{2\pi\sigma^2}\int_R e^{-\frac{1+\lambda}{2\sigma^2}(\beta-\frac{y}{1+\lambda})^2}d\beta\; e^{-\frac{\lambda y^2}{2\sigma^2(1+\lambda)}}=\sqrt{\frac{\lambda}{\lambda+1}}\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{y^2}{2\sigma^2}\frac{\lambda}{\lambda+1}}$$

Wobec tego rozkład a posteriori wektora β jest postaci

$$f(\beta)=\frac{g(y,\beta)}{k(y)}=\frac{\sqrt{1+\lambda}}{\sqrt{2\pi}\sigma}e^{-\frac{1+\lambda}{2\sigma^2}\left(\beta-\frac{y}{1+\lambda}\right)^2}$$

Moda rozkładu a posteriori (w tym przypadku także wartość oczekiwana) jest estymatorem bayesowskim parametru β.

Regresja lasso

Przyjmijmy warunkowy rozkład obserwacji $Y\mid\beta\sim N(\beta,\sigma^2)$ i rozkład Laplace'a jako *a priori* parametru $\beta\sim\mathcal{L}_a(\frac{\lambda}{2\sigma^2})$. Wobec tego $p(y\mid\beta)=\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(y-\beta)^2}{2\sigma^2}}$ i $h(\beta)=\frac{\lambda}{4\sigma^2}e^{-\frac{\lambda}{2\sigma^2}|\beta|}$. Łączny rozkład wektora (y,β) jest postaci

$$g(y,\beta)=p(y\mid\beta)h(\beta)=\frac{\lambda}{4\sqrt{2\pi\sigma^3}}e^{-\frac{1}{2\sigma^2}[(y-\beta)^2+\lambda|\beta|]}$$ i w konsekwencji

$$f(\beta)=\frac{g(y,\beta)}{\int_R g(y,\beta)d\beta}=C(y,\lambda,\sigma)e^{-\frac{1}{2\sigma^2}[(y-\beta)^2+\lambda|\beta|]}.$$

Moda rozkładu *a posteriori* czyli $\arg\max_\beta k(\beta)=\arg\min_\beta[(y-\beta)^2+\lambda\mid\beta\mid]$ jest estymatorem lasso w rozważanym problemie.

Do zebrania danych dotyczących próbek drobno posiekanego czystego mięsa wykorzystano analizator żywności i pasz Tecator Infratec pracujący w zakresie długości fal od 850 do 1050 nm na zasadzie transmisji w bliskiej podczerwieni (NIT). Zmierzono 215 próbek. W każdej próbce mierzono zawartość tłuszczu oraz 100-kanałowe widmo absorbancji. Ponieważ oznaczanie zawartości tłuszczu za pomocą chemii analitycznej jest czasochłonne, chcielibyśmy zbudować model umożliwiający przewidywanie zawartości tłuszczu w nowych próbkach przy użyciu 100 absorbancji, które można łatwiej zmierzyć.

Plik faraway9.new

```
library(faraway)
data (meatspec)#Spektrometria mięsa do oznaczania zawartości tłuszczu
#zmienne V1-V100 - absorpcja w zakresie 100 długości fali
# 101-sza zmienna fat zawiera dane dotyczące tłuszczu
# pierwszych 172 obserwacji potraktujemy jako zbiór uczący a resztę 173 d0 215 tako zbiór testowy
help("meatspec")
model1 <- lm (fat~.,meatspec[1:172,]) # dopasowujemy model pełny na zbiorze uczącym
summary(model1)$r.squared
rmse <- function(x,y) sqrt(mean((x-y)^2))# funkcja wyznaczająca RMSE
rmse (model1$fit, meatspec$fat [1:172]) # rms dla zbioru uczącego
        0.6903167
rmse (predict(model1,meatspec[173:215,]),meatspec$fat[173:215])# rms dla zbioru testowego
```

3.814

Widać że wyestymowany model ma słabe własności predykcyjne

```
mode12 <-step(model1) #regresja krokowa dla zbioru uczącego
```
Model regresji krokowej usunął 28 spośród 100 zmiennych

```
rmse(mode12$fit,meatspec$fat[1:172])
```
0.7095069
```
rmse(predict(mode12,meatspec[173:215,]),meatspec$fat[173:215])
```
3.590245

Ten model ma także słabe własności predykcyjne

```
#Regresja grzbietowa
library (MASS)
yc <- meatspec$fat[1:172]-mean(meatspec$fat[1:172])
mm <- apply(meatspec[1:172,-101],2,mean)# wyznaczamy średnie dla 100 kolumn zbioru uczącego
trainx <- as.matrix(sweep(meatspec[1:172,-101],2,mm))#centrowanie predyktorów
gridge <- lm.ridge(yc~trainx,lambda=seq(0,5e-8,1e-9))
matplot(gridge$lambda,t(gridge$coef), type="l",lty=1,xlab=expression (lambda), ylab=expression (hat
(beta)))
select (gridge)
abline (v=1.8e-8)
which.min(gridge$GCV) #wyznaczenie optymalnych współczynników  metodą GCV
ypredg <- scale(trainx,center=FALSE,scale=gridge$scales)%*% gridge$coef [, 19] +
mean(meatspec$fat[1:172])
rmse(ypredg,meatspec$fat[1:172])
testx <- as.matrix(sweep (meatspec [173:215,-101], 2, mm))
ytpredg <- scale(testx,center=FALSE, scale=gridge$scales)%*% gridge$coef [, 19] +
mean(meatspec$fat[1:172])
rmse (ytpredg, meatspec$fat[173:215])

# duży błąd jest tylko w przypadku jednej obserwacji
ytpredg- meatspec$fat[173:215]
c(ytpredg[13],meatspec$fat [172+13])
#po jej usunięciu rmse jest całkiem dobry
rmse(ytpredg[-13],meatspec$fat[173:215][-13])



library(glmnet)
require(graphics)
#  zobacz     https://www.statology.org/lasso-regression-in-r/

summary(meatspec)
#To perform lasso regression, we'll use functions from the glmnet package
#This package requires the response variable to be a vector and the set of predictor variables
#to be of the class data.matrix.


#define response variable
y <- meatspec[1:172,101]

#define matrix of predictor variables
x <- data.matrix(meatspec[1:172,1:100])

#perform k-fold cross-validation to find optimal lambda value
```

```
cv_model <- cv.glmnet(x, y, alpha = 1)
cv_model_ridge <- cv.glmnet(x, y, alpha = 0)
#find optimal lambda value that minimizes test MSE
best_lambda <- cv_model$lambda.min
best_lambda
best_lambda_ridge <- cv_model_ridge$lambda.min
best_lambda_ridge



#produce plot of test MSE by lambda value
plot(cv_model)
plot(cv_model_ridge)



#find coefficients of best model

#Note that setting alpha equal to 0 is equivalent to using ridge regression
#and setting alpha to some value between 0 and 1 is equivalent to using an elastic net.
best_model <- glmnet(x, y, alpha = 1, lambda = best_lambda)
coef(best_model)
#define new observation
#new = as.matrix(meatspec[173:225,1:100]), nrow=1, ncol=100))
new = data.matrix(meatspec[173:215,1:100])

#use lasso regression model to predict response value
predict(best_model, s = best_lambda, newx = new)

#use fitted best model to make predictions
y_predicted <- predict(best_model, s = best_lambda, newx = new)

#find SST and SSE
y_test<-meatspec[173:215,101]

sst <- sum((y_test - mean(y_test))^2)
sse <- sum((y_predicted - y_test)^2)

#find R-Squared
rsq <- 1 - sse/sst
rsq

rmse (y_predicted, meatspec$fat[173:215])
y_predicted- meatspec$fat[173:215]

best_model_ridge <- glmnet(x, y, alpha = 0, lambda = best_lambda_ridge)
coef(best_model_ridge)
#use ridge regression model to predict response value
predict(best_model_ridge, s = best_lambda_ridge, newx = new)
#use fitted best model to make predictions
y_predicted_ridge <- predict(best_model_ridge, s = best_lambda_ridge,newx = new)
sse <- sum((y_predicted_ridge - y_test)^2)
#find R-Squared
rsq <- 1 - sse/sst
rsq
rmse (y_predicted_ridge, meatspec$fat[173:215])
```

LASSO plik lasso.R

```
# zobacz https://www.statology.org/lasso-regression-in-r/
library(glmnet)
require(graphics)
help("mtcars")
pairs(mtcars, main = "mtcars data", gap = 1/4)
coplot(mpg ~ disp | as.factor(cyl), data = mtcars,
       panel = panel.smooth, rows = 1)
# possibly more meaningful, e.g., for summary() or bivariate plots:
mtcars2 <- within(mtcars, {
  vs <- factor(vs, labels = c("V", "S"))
  am <- factor(am, labels = c("automatic", "manual"))
  cyl  <- ordered(cyl)
  gear <- ordered(gear)
  carb <- ordered(carb)
})
summary(mtcars2)
#To perform lasso regression, we'll use functions from the glmnet package
#This package requires the response variable to be a vector and the set of predictor variables
#to be of the class data.matrix.


#define response variable
y <- mtcars$hp

#define matrix of predictor variables
x <- data.matrix(mtcars[, c('mpg', 'wt', 'drat', 'qsec')])

#perform k-fold cross-validation to find optimal lambda value
cv_model <- cv.glmnet(x, y, alpha = 1)
cv_model_ridge <- cv.glmnet(x, y, alpha = 0)
#find optimal lambda value that minimizes test MSE
best_lambda <- cv_model$lambda.min
best_lambda
best_lambda_ridge <- cv_model_ridge$lambda.min
best_lambda_ridge



#produce plot of test MSE by lambda value
plot(cv_model)
plot(cv_model_ridge)



#find coefficients of best model

#Note that setting alpha equal to 0 is equivalent to using ridge regression
#and setting alpha to some value between 0 and 1 is equivalent to using an elastic net.
best_model <- glmnet(x, y, alpha = 1, lambda = best_lambda)
coef(best_model)
#define new observation
new = matrix(c(24, 2.5, 3.5, 18.5), nrow=1, ncol=4)

#use lasso regression model to predict response value
predict(best_model, s = best_lambda, newx = new)
```

```
#use fitted best model to make predictions
y_predicted <- predict(best_model, s = best_lambda, newx = x)

#find SST and SSE
sst <- sum((y - mean(y))^2)
sse <- sum((y_predicted - y)^2)

#find R-Squared
rsq <- 1 - sse/sst
rsq
```

To samo dla regresji grzbietowej

```
best_model_ridge <- glmnet(x, y, alpha = 0, lambda = best_lambda_ridge)
coef(best_model_ridge)
#use ridge regression model to predict response value
predict(best_model_ridge, s = best_lambda_ridge, newx = new)
#use fitted best model to make predictions
y_predicted_ridge <- predict(best_model_ridge, s = best_lambda_ridge, newx = x)
sse <- sum((y_predicted_ridge - y)^2)
#find R-Squared
rsq <- 1 - sse/sst
rsq
```

Zbiór Hitters to ramka danych z obserwacjami dwudziestu zmiennych dla 322 głównych graczy ligowych w sezonie 1986-87.

Zmienne

- AtBat- liczba uderzeń kijem w 1986 roku
- Hits- liczba trafień kijem w 1986 roku
- HmRun -liczba uderzeń (home runs) w 1986 roku, które pozwalają pałkarzowi wykonać pełne okrążenie baz bez zatrzymywania się i zdobyć obieg.
- Runs- liczba biegów w 1986 roku
- RBI- liczba biegów *batted in* w 1986 roku
- Walks- liczba *spacerów walks* w 1986 roku
- Years- liczba lat w głównej lidze
- CAtBat- liczba uderzeń kijem trakcie całej kariery
- CHits-liczba trafień kijem w trakcie kariery
- CHmRun-liczba *home runs* w trakcie kariery
- CRuns- liczba biegów w trakcie kariery
- CRBI-liczba biegów *batted in* w trakcie kariery
- CWalks- liczba *spacerów walks* w trakcie kariery
- League- zmienna czynnikowa typu ligi A lub N na koniec 1986 roku
- Division-zmiena czynnikowa z poziomami E i W
- PutOuts-liczba autów w 1986 roku
- Assists-liczba asyst w 1986 roku
- Errors- liczba błedów w 1986 roku
- Salary-zarobki w 1987 roku w tys. dolarów
- NewLeague- zmienna czynnikowa z poziomami A i N oznaczające ligę gracza na początku 1987 roku

Plik JWHT

```
library(ISLR2)
View(Hitters)
names(Hitters)
dim(Hitters)
#Sprawdzenie czy są wiersze z brakującymi danymi
sum(is.na(Hitters$Salary))
#usunięcie wierszy z brakami
Hitters <- na.omit(Hitters)
dim(Hitters)
sum(is.na(Hitters))
#Wybór najlepszego podzbioru zmiennych objaśniających
library(leaps)
regfit.full <- regsubsets(Salary~., Hitters)
summary(regfit.full)
regfit.full <- regsubsets(Salary~ ., data = Hitters ,nvmax = 19)
reg.summary <- summary(regfit.full)
```

```
names(reg.summary)
reg.summary$rsq
par(mfrow = c(2, 2))
plot(reg.summary$rss , xlab = "Number of Variables", ylab = "RSS", type = "l")
plot(reg.summary$adjr2 , xlab = "Number of Variables",ylab = "Adjusted RSq", type = "l")
which.max(reg.summary$adjr2)
points (11, reg.summary$adjr2 [11] , col = "red", cex = 2,pch = 20)

plot(reg.summary$cp, xlab = "Number of Variables",ylab = "Cp", type = "l")
which.min(reg.summary$cp)
points (10, reg.summary$cp[10] , col = "red", cex = 2,pch = 20)
which.min(reg.summary$bic)
plot(reg.summary$bic , xlab = "Number of Variables", ylab = "BIC", type = "l")
points (6, reg.summary$bic [6], col = "red", cex = 2, pch = 20)

plot(regfit.full , scale = "r2")
plot(regfit.full , scale = "adjr2")
plot(regfit.full , scale = "Cp")
plot(regfit.full , scale = "bic")
coef(regfit.full , 6)


regfit.fwd <- regsubsets(Salary ~ ., data = Hitters , nvmax = 19, method = "forward")
summary(regfit.fwd)
regfit.bwd <- regsubsets(Salary ~ ., data = Hitters ,nvmax = 19, method = "backward")
summary(regfit.bwd)
coef(regfit.full , 7)
coef(regfit.fwd , 7)
coef(regfit.bwd , 7)

set.seed (1)
train <- sample(c(TRUE , FALSE), nrow(Hitters),replace = TRUE)
test <- (!train)
regfit.best <- regsubsets(Salary ~ .,data = Hitters[train , ], nvmax = 19)
test.mat <- model.matrix(Salary ~ ., data = Hitters[test , ])


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)
}
val.errors
which.min(val.errors)
coef(regfit.best , 7)

predict.regsubsets <- function(object , newdata , id, ...) {
   form <- as.formula(object$call [[2]])
   mat <- model.matrix(form , newdata)
   coefi <- coef(object , id = id)
   xvars <- names(coefi)
   mat[, xvars] %*% coefi
  }

regfit.best <- regsubsets(Salary~ ., data = Hitters ,nvmax = 19)
coef(regfit.best , 7)
k <- 10
n <- nrow(Hitters)
set.seed (1)
```

```
folds <- sample(rep (1:k, length = n))
cv.errors <- matrix(NA, k, 19,dimnames = list(NULL , paste (1:19)))
for (j in 1:k) {
  best.fit <- regsubsets(Salary ~ .,data = Hitters[folds != j, ],nvmax = 19)
  for (i in 1:19) {
    pred <- predict(best.fit , Hitters[folds == j, ], id = i)
    cv.errors [j, i] <- mean (( Hitters$Salary[folds == j] - pred)^2)
    }
}
mean.cv.errors <- apply(cv.errors , 2, mean)
mean.cv.errors
par(mfrow = c(1, 1))
plot(mean.cv.errors , type = "b")
reg.best <- regsubsets(Salary ~ ., data = Hitters , nvmax = 19)
coef(reg.best , 10)


#Ridge  regression and the LASSO

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


library(glmnet)
grid <- 10^seq(10, -2, length = 100)
ridge.mod <- glmnet(x, y, alpha = 0, lambda = grid)
dim(coef(ridge.mod))
ridge.mod$lambda[50]
coef(ridge.mod)[, 50]
sqrt(sum(coef(ridge.mod)[-1, 50]^2))
ridge.mod$lambda[60]
coef(ridge.mod)[, 60]
sqrt(sum(coef(ridge.mod)[-1, 60]^2))
predict(ridge.mod , s = 50, type = "coefficients")[1:20, ]
set.seed (1)
train <- sample (1: nrow(x), nrow(x) / 2)
test <- (-train)
y.test <- y[test]

ridge.mod <- glmnet(x[train , ], y[train], alpha = 0,lambda = grid , thresh = 1e-12)
ridge.pred <- predict(ridge.mod , s = 4, newx = x[test , ])
mean (( ridge.pred - y.test)^2)
mean (( mean(y[train ]) - y.test)^2)
ridge.pred <- predict(ridge.mod , s = 1e10 , newx = x[test , ])
mean (( ridge.pred - y.test)^2)

ridge.pred <- predict(ridge.mod , s = 0, newx = x[test , ],exact = T, x = x[train , ], y = y[train ])
mean (( ridge.pred - y.test)^2)
lm(y ~ x, subset = train)
predict(ridge.mod , s = 0, exact = T, type = "coefficients",x = x[train, ], y = y[train])[1:20, ]
set.seed (1)
cv.out <- cv.glmnet(x[train , ], y[train], alpha = 0)
plot(cv.out)
bestlam <- cv.out$lambda.min
bestlam
ridge.pred <- predict(ridge.mod , s = bestlam , newx = x[test , ])
mean (( ridge.pred - y.test)^2)
out <- glmnet(x, y, alpha = 0)
predict(out , type = "coefficients", s = bestlam)[1:20, ]
```

```
lasso.mod <- glmnet(x[train , ], y[train], alpha = 1,lambda = grid)
plot(lasso.mod)
set.seed (1)
cv.out <- cv.glmnet(x[train , ], y[train], alpha = 1)
plot(cv.out)
bestlam <- cv.out$lambda.min
lasso.pred <- predict(lasso.mod , s = bestlam , newx = x[test , ])
mean (( lasso.pred - y.test)^2)

out <- glmnet(x, y, alpha = 1, lambda = grid)
lasso.coef <- predict(out , type = "coefficients",s = bestlam)[1:20, ]
lasso.coef

library(pls)
set.seed (2)
pcr.fit <- pcr(Salary ~ ., data = Hitters , scale = TRUE , validation = "CV")
summary(pcr.fit)
validationplot(pcr.fit , val.type = "MSEP")
set.seed (1)
pcr.fit <- pcr(Salary ~ ., data = Hitters , subset = train ,scale = TRUE , validation = "CV")
validationplot(pcr.fit , val.type = "MSEP")
pcr.pred <- predict(pcr.fit , x[test , ], ncomp = 5)
mean (( pcr.pred - y.test)^2)
pcr.fit <- pcr(y ~ x, scale = TRUE , ncomp = 5)
summary(pcr.fit)

set.seed (1)
pls.fit <- plsr(Salary ~ ., data = Hitters , subset = train ,scale = TRUE , validation = "CV")
summary(pls.fit)
pls.pred <- predict(pls.fit , x[test , ], ncomp = 1)
mean (( pls.pred - y.test)^2)
pls.fit <- plsr(Salary ~ ., data = Hitters , scale = TRUE , ncomp = 1)
summary(pls.fit)
```