107.330 Statistische Simulation und computerintensive Methoden

Shrinkage Methods for Regression

Alexandra Posekany

WS 2020

Motivation

We start out from the multiple linear model

$$y = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p + \epsilon,$$

where we assume now we have a large amount of possible predictors $x_i, \ldots, x_p$ and consider a nested model

$$y = \beta_0 + \beta_{i_1} x_{i_1} + \ldots + \beta_{i_k} x_{i_k} + \epsilon,$$

where $k < p$ and $i_1, \ldots, i_k$ is a subset of $1, \ldots, p$.

Model selection approaches

Typically, one selects a model which is a subset out of the q predictors by one of several standard approaches:

- Testing the model coefficients independently to be equal to 0

  Student's t test is included in the R summary output

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	49.99303171	6.18640648	8.0811101	4.312200e-10
Fertility	-0.52070092	0.07868790	-6.6172931	5.139985e-08
Agriculture	-0.22879685	0.03906287	-5.8571441	6.374386e-07
Catholic	0.08333381	0.02178681	3.8249658	4.275083e-04
Infant.Mortality	0.28436994	0.30040325	0.9466274	3.492434e-01

This has the disadvantage that the influence of removing the least relevant explanatory variable on the other model coefficients and their significance is disregarded.

Model selection approaches

A different notion of model selection is given by step-wise model selection based on **information criteria** by minimising said criteria.

- Akaike Information Criterion (AIC)

$$\mathrm{AIC} = -2\ln(\hat{L}(\theta)) + 2k$$

Because this is the negative log-likelihood function penalised by the number of model parameters q minimising the AIC is equivalent to maximising the likelihood function which is the statistical basis for contruction of many tests and models.

For linear regression AIC is equivalent to Mallow's C_p.

- Bayesian Information Criterion (BIC)

$$\mathrm{BIC} = -2\ln(\hat{L}(\theta)) + ln(n)k$$

Assumes a prior probability of each candidate model $\frac{1}{\#\text{candidate models}}$ which is combined with the likelihood function which comprises the information of the data. BIC puts a larger penalty on model size (for n>7) than AIC and thus selects smaller models.

Information Criteria based Model selection

```
> step(lm(Education~.-Examination,data=swiss))
Start:  AIC=160.13
Education ~ (Fertility + Agriculture + Examination + Catholic +
    Infant.Mortality) - Examination

                  Df Sum of Sq    RSS    AIC
- Infant.Mortality  1     24.46 1170.8 159.12
<none>                         1146.3 160.13
- Catholic          1    399.32 1545.7 172.17
- Agriculture       1    936.34 2082.7 186.19
- Fertility         1   1195.15 2341.5 191.69

Step:  AIC=159.12
Education ~ Fertility + Agriculture + Catholic

              Df Sum of Sq    RSS    AIC
<none>                     1170.8 159.12
- Catholic     1    410.71 1581.5 171.25
- Agriculture  1   1079.89 2250.7 187.84
- Fertility    1   1289.36 2460.2 192.02

Call:
lm(formula = Education ~ Fertility + Agriculture + Catholic,
    data = swiss)

Coefficients:
(Intercept)    Fertility  Agriculture     Catholic
    53.8505      -0.4888      -0.2380       0.0844
```

Regularised approaches

An alternative to applying a model selection method after the actual model fit in order to limit the parameters from p down to k is to fit a model including all p predictors, but **regularizing** the β_i coefficients during the process. The goal is the "**shrink**" them towards zero and thus effectively removing parameters with small coefficients from the model.

The best known shrinkage methods for regression are

- **Ridge regression** and
- the **LASSO**.

Ridge regression

Consider a sample of size n, the ridge regression coefficient vector β^{Ridge} minimizes

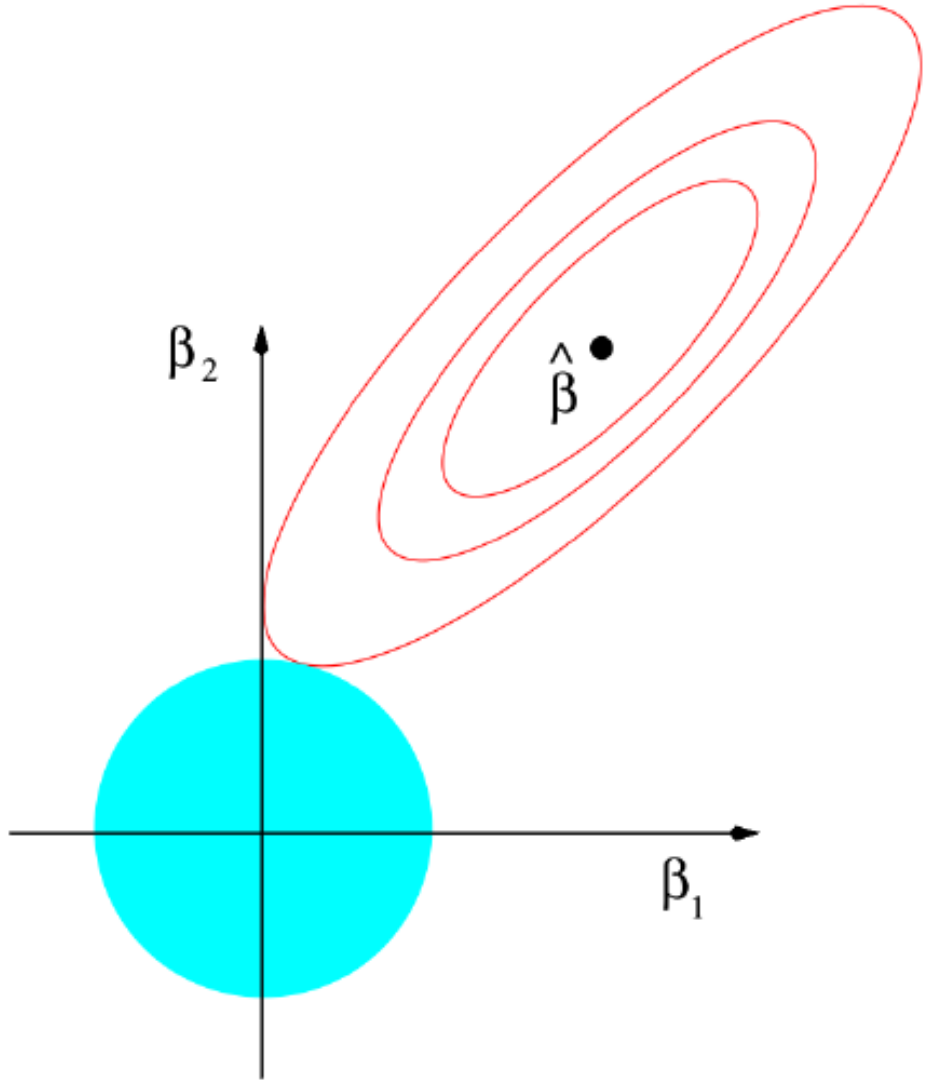
$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right) + \lambda\sum_{j=1}^{p}\beta_j^2,$$

where $\lambda \geq 0$ is a tuning parameter which needs to be decided upon separately.

Remarks:

- λ is called the **shrinkage parameter**
- λ serves to control the relative impact of the regression coefficients. If $\lambda = 0$ there is no shrinkage and one obtains the normal LS fit. If $\lambda \to \infty$ then the impact grows and the coefficients tend to zero.
- The intercept is not included in the shrinkage process.

Graphical Motivation of the Ridge Regression penalty



Further remarks about ridge regression

- Denoting $\mathbf{X}$ now as the $n \times p$ matrix contain the p predictors (no columns of 1), then

$$\beta^{Ridge} = \left(\mathbf{X}^\top \mathbf{X} - \lambda \mathbf{I}_p\right)^{-1} \mathbf{X}^\top \mathbf{y}$$

  (plus β_0 estimated separately).
- Because the shrinkage part has a stabilising effect $\mathbf{X}^\top \mathbf{X}$ does not need to be of full rank!
- For every λ a different β^{Ridge} is obtained, therefore the crucial value of λ chosen using cross validation.
- fFr ridge regression the scales of the individual predictors matter and therefore it is usually assumed that all predictors are standardised to have mean 0 and variance 1. This implies that β_0 is the mean of y.
- Compared to LS, ridge regression includes bias but reduces in turn the variance and is able to deal with dependence of the regressors.

LASSO

The ridge regression estimator has one big disadvantage - it will basically never estimate the coefficients of the non-relevant predictors as zero but just shrink them towards zero.

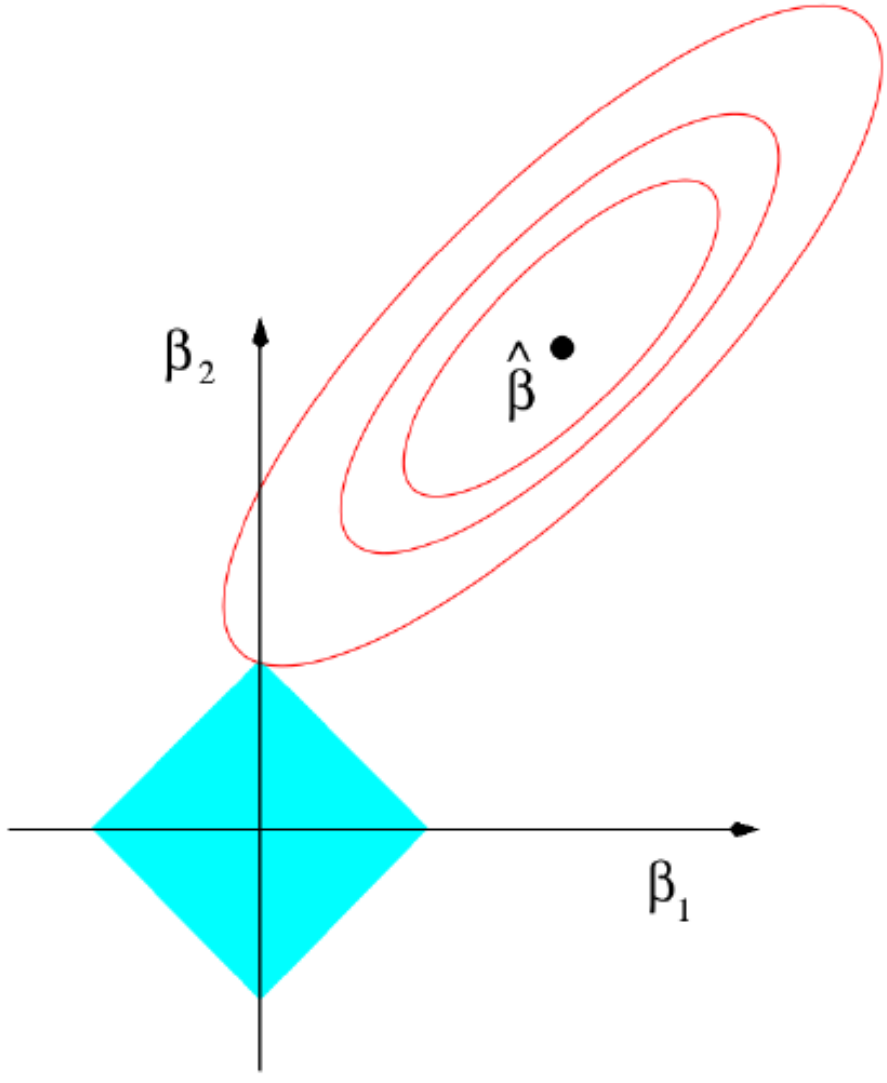
The **lasso** was developed to overcome this issue.

Using the same setup as for the ridge regression, lasso regression coefficient vector β^{LASSO} minimizes

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

Thus, the ridge estimator uses an L_2 penalty while the lasso an L_1 penalty. The advantage is that the lasso also also shrinks the coefficients towards zero but it also **forces** some to be **zero**.

Graphical Motivation of the LASSO Regression penalty



Remarks about the lasso

- Due to the construction of the LASSO it is possible to reduce coefficients actually to 0.
- The tuning parameter λ needs to be again selected separately, usually using CV.
- The lasso can be seen as a continues subset selection method.
- The lasso penalty makes the constraints non-linear in y and therefore is more difficult to compute.
- LASSO stands for **least absolute shrinkage and selection operator**.

Example for Ridge Regression and LASSO in R

```
> library(glmnet)
Loading required package: Matrix
Loading required package: foreach
Loaded glmnet 2.0-18
> set.seed(1); n <- 500; p=20
> X <- matrix(rnorm(n*p), ncol=p)
> X <- scale(X)
> eps <- rnorm(n, sd=5)
> beta <- 3:5
> y <-  2 + X[,1:3]  %*% beta + eps
```

Ridge Regression in R

The connection between linear regression and Ridge regression is via the penalty term of the squared coefficients.

```
> fitLM <- lm(y~X)
> round(coef(fitLM),2)
(Intercept)          X1          X2          X3          X4          X5
       2.01        3.04        3.94        4.92       -0.12        0.24
         X6          X7          X8          X9         X10         X11
      -0.12       -0.16       -0.15       -0.06       -0.17       -0.08
        X12         X13         X14         X15         X16         X17
      -0.14        0.43       -0.08       -0.03        0.02        0.00
        X18         X19         X20
      -0.21       -0.18       -0.31
> # root sum of squared coefficients
> sqrt(sum(coef(fitLM)[-1]^2))
[1] 7.044342
```

Ridge Regression in R

The package **glmnet** can perform ridge regression in R using the function `glmnet`. To obtain a ridge regression there, the parameter `alpha` needs to be set to zero.

```
> lambda.grid <- 10^seq(10, -2,length=100)
> fitRR <- glmnet(x=X, y=y, alpha=0, lambda=lambda.grid)
> dim(coef(fitRR))
[1]  21 100
```

Lasso in R

The function `glmnet` can also compute the lasso. For that purpose the parameter `alpha` needs to be set to one.

```
> fitL <- glmnet(x=X, y=y, alpha=1, lambda=lambda.grid)
> dim(coef(fitL))
[1]  21 100
```

Ridge Regression in R II

```
>
> fitRR$lambda[50]
[1] 11497.57
> round(coef(fitRR)[,50],2)
(Intercept)          V1          V2          V3          V4
       2.01        0.00        0.00        0.00        0.00
         V6          V7          V8          V9         V10
       0.00        0.00        0.00        0.00        0.00
        V12         V13         V14         V15         V16
       0.00        0.00        0.00        0.00        0.00
        V18         V19         V20
       0.00        0.00        0.00
> sqrt(sum(coef(fitRR)[-1,50]^2))
[1] 0.004567586
```

Lasso in R II

Unlike Ridge regression LASSO shrinks the irrelevant coefficients to 0.

```
>
> fitL$lambda[50]
[1] 11497.57
> round(coef(fitL)[,50],2)
(Intercept)          V1          V2          V3          V4          V5
       2.01        0.00        0.00        0.00        0.00        0.00
         V6          V7          V8          V9         V10         V11
       0.00        0.00        0.00        0.00        0.00        0.00
        V12         V13         V14         V15         V16         V17
       0.00        0.00        0.00        0.00        0.00        0.00
        V18         V19         V20
       0.00        0.00        0.00
> sqrt(sum(coef(fitL)[-1,50]^2))
[1] 0
```

Ridge Regression in R III

Ridge regression regularises coefficients but provides less shrinkage than LASSO.

```
> fitRR$lambda[70]
[1] 43.28761
> round(coef(fitRR)[,70],2)
(Intercept)          V1          V2          V3          V4          V5
       2.01        0.44        0.55        0.72        0.02        0.06
         V6          V7          V8          V9         V10         V11
       0.02        0.02       -0.02        0.00       -0.06        0.00
        V12         V13         V14         V15         V16         V17
      -0.02        0.12        0.01       -0.04        0.06       -0.05
        V18         V19         V20
       0.03        0.01       -0.07
> sqrt(sum(coef(fitRR)[-1,70]^2))
[1] 1.028724
```

Lasso in R III

Unlike Ridge regression LASSO shrinks more irrelevant coefficients to 0.

```
> fitL$lambda[70]
[1] 43.28761
> round(coef(fitL)[,70],2)
(Intercept)          V1          V2          V3          V4          V5
       2.01        0.00        0.00        0.00        0.00        0.00
         V6          V7          V8          V9         V10         V11
       0.00        0.00        0.00        0.00        0.00        0.00
        V12         V13         V14         V15         V16         V17
       0.00        0.00        0.00        0.00        0.00        0.00
        V18         V19         V20
       0.00        0.00        0.00
> sqrt(sum(coef(fitL)[-1,70]^2))
[1] 0
```

Ridge Regression in R IV

Coefficients of relevance should remain in the model. Ridge regression provides less shrinkage and has larger coefficient values, as all coefficients are non-zero.

```
> fitRR$lambda[90]
[1] 0.1629751
> round(coef(fitRR)[,90],2)
(Intercept)          V1          V2          V3          V4          V5
       2.01        2.98        3.85        4.82       -0.11        0.24
         V6          V7          V8          V9         V10         V11
      -0.12       -0.15       -0.15       -0.06       -0.17       -0.07
        V12         V13         V14         V15         V16         V17
      -0.14        0.43       -0.07       -0.03        0.03       -0.01
        V18         V19         V20
      -0.20       -0.17       -0.31
> sqrt(sum(coef(fitRR)[-1,90]^2))
[1] 6.887053
```

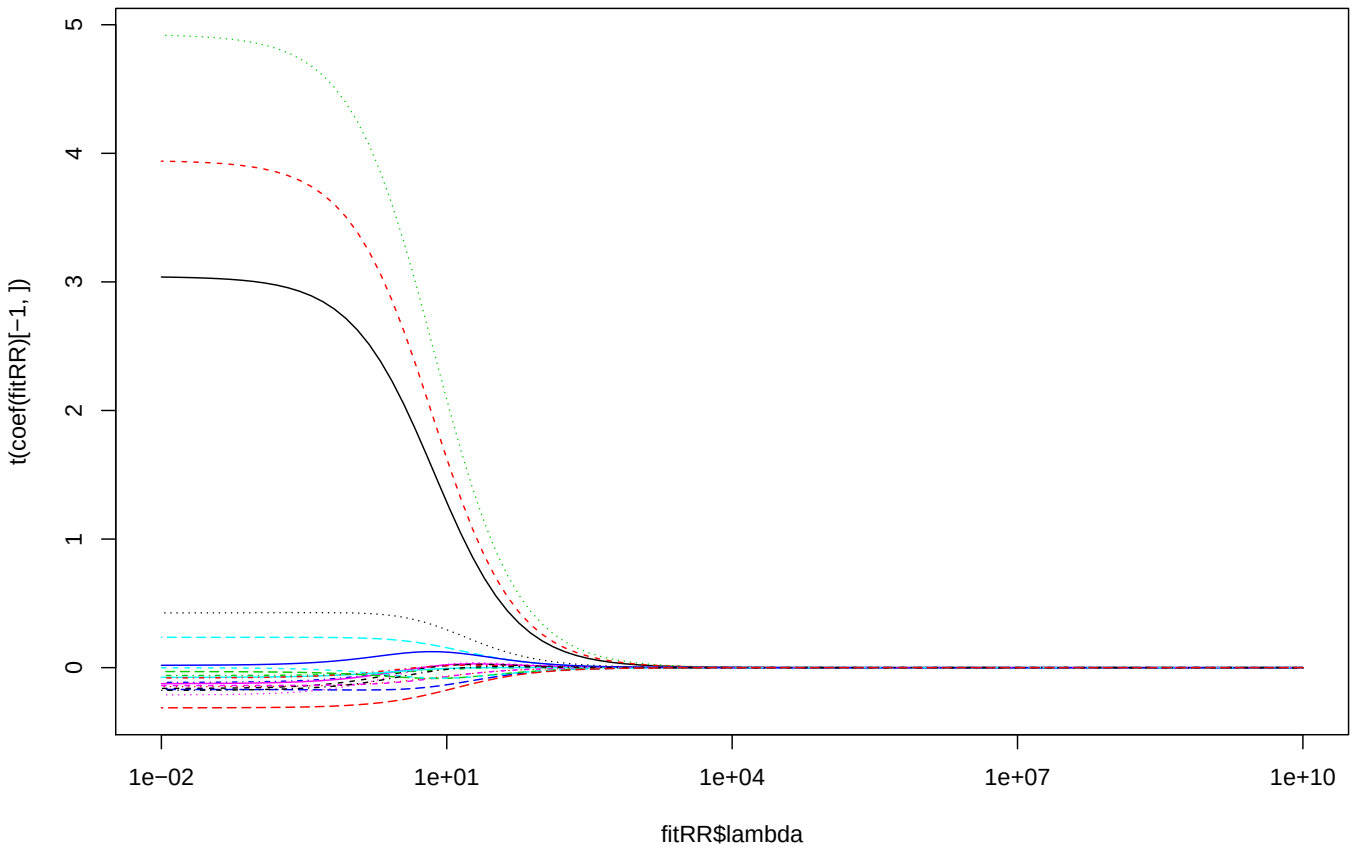
Lasso in R IV

LASSO provides more shrinkage, shrinking irrelevant coefficients to 0 and keeping only the 3 variables which were actually used to generate the y data with non-zero coefficients. Here is less noise than with Ridge regression, but more bias.

```
> fitL$lambda[90]
[1] 0.1629751
> round(coef(fitL)[,90],2)
(Intercept)          V1          V2          V3          V4          V5
       2.01        2.88        3.75        4.72        0.00        0.09
         V6          V7          V8          V9         V10         V11
       0.00        0.00        0.00        0.00        0.00        0.00
        V12         V13         V14         V15         V16         V17
       0.00        0.30        0.00        0.00        0.00        0.00
        V18         V19         V20
      -0.01       -0.03       -0.11
> sqrt(sum(coef(fitL)[-1,90]^2))
[1] 6.686686
```

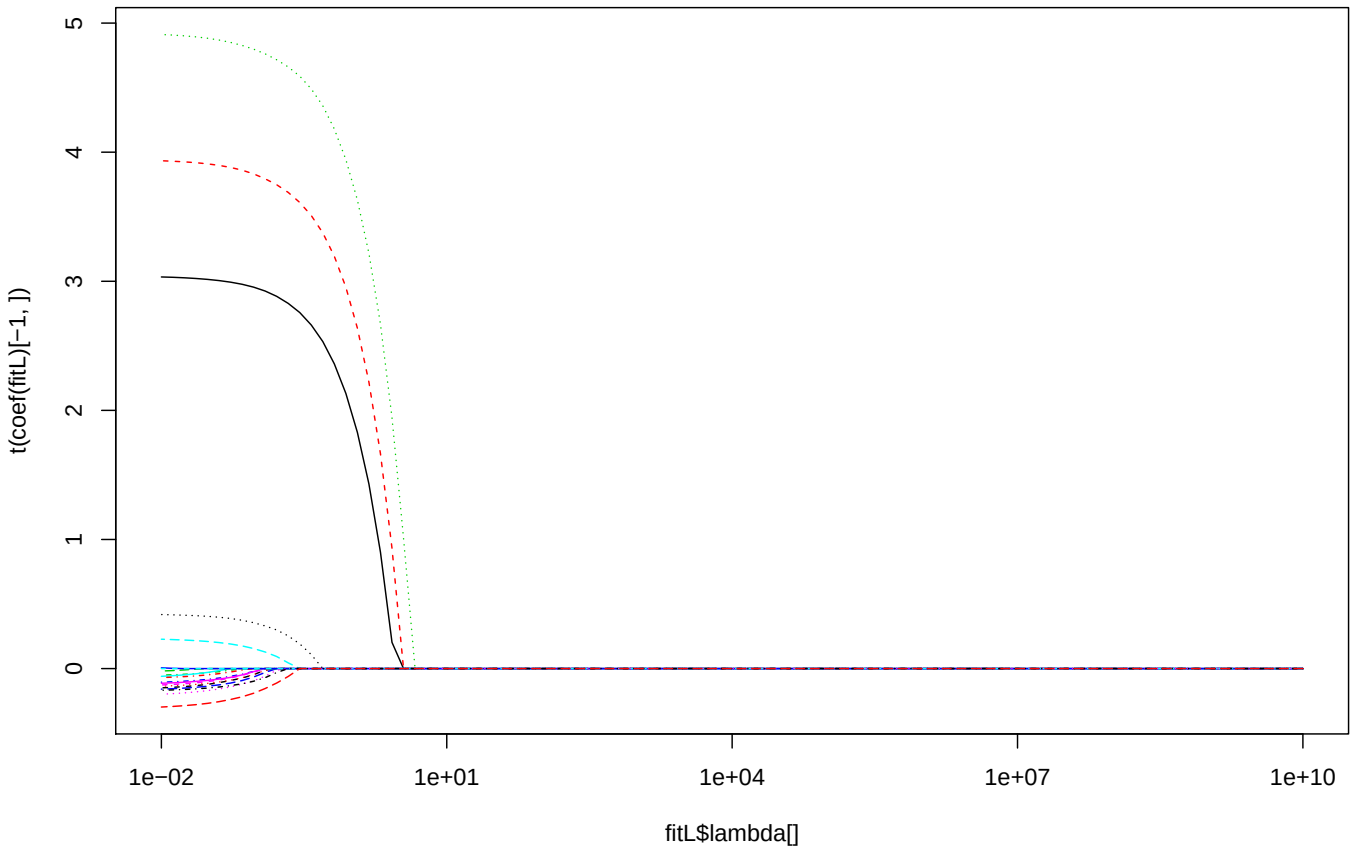
Ridge Regression in R V

```
> matplot(fitRR$lambda, t(coef(fitRR)[-1,]), type="l",log="x")
```



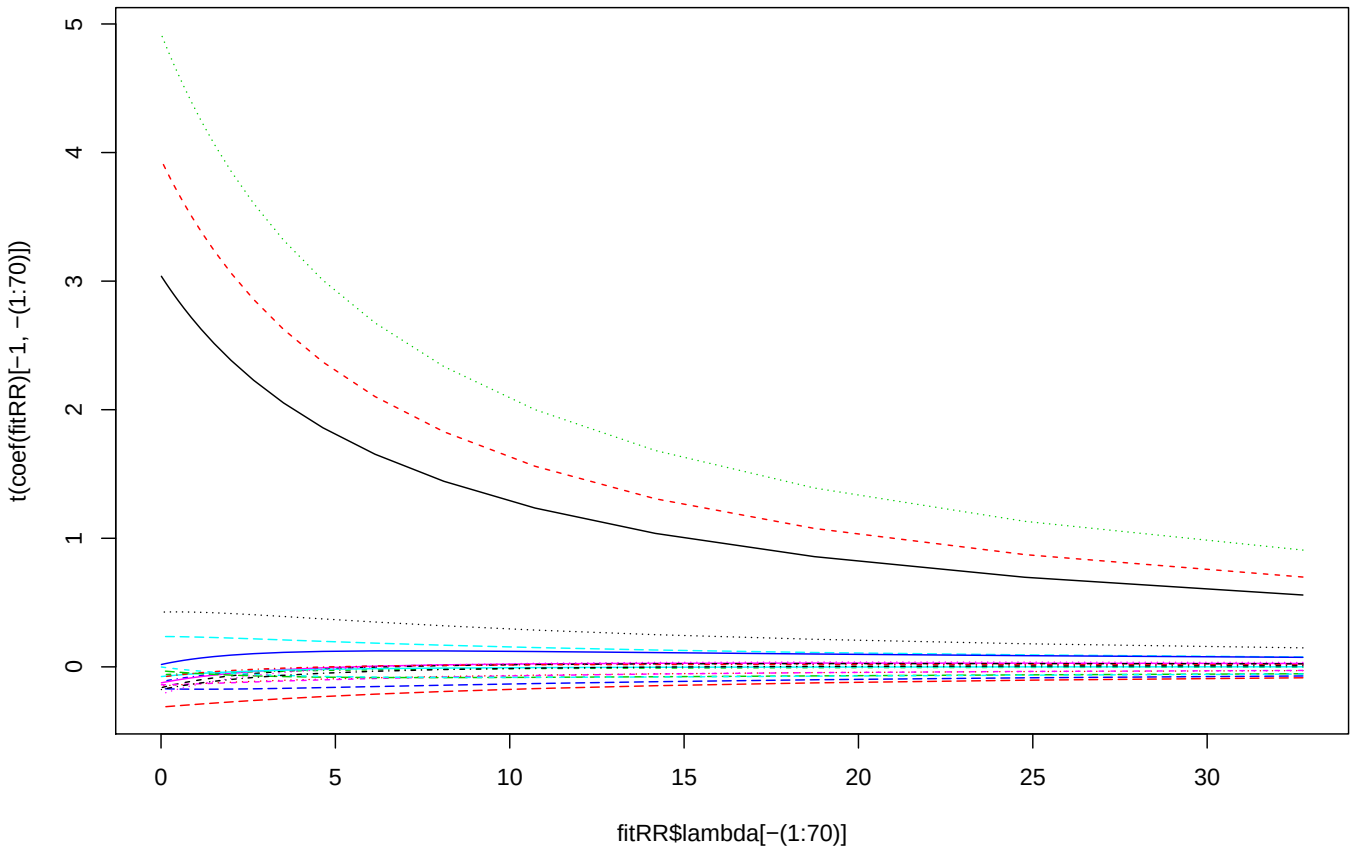
Lasso in R V

```
> matplot(fitL$lambda[], t(coef(fitL)[-1,]), type="l",log="x")
```



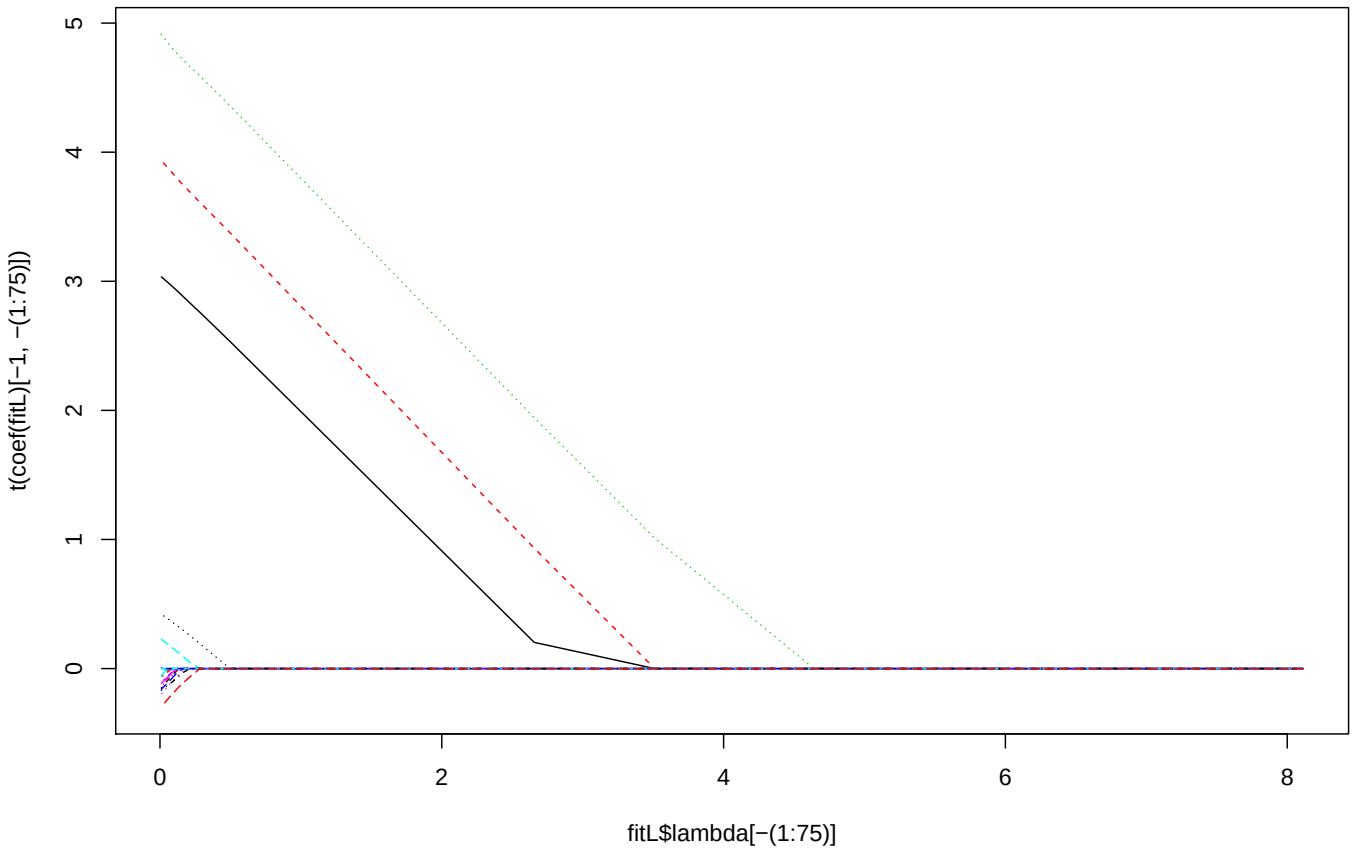
Ridge Regression in R VI

```
> matplot(fitRR$lambda[-(1:70)], t(coef(fitRR)[-1,-(1:70)]), type="l")
```



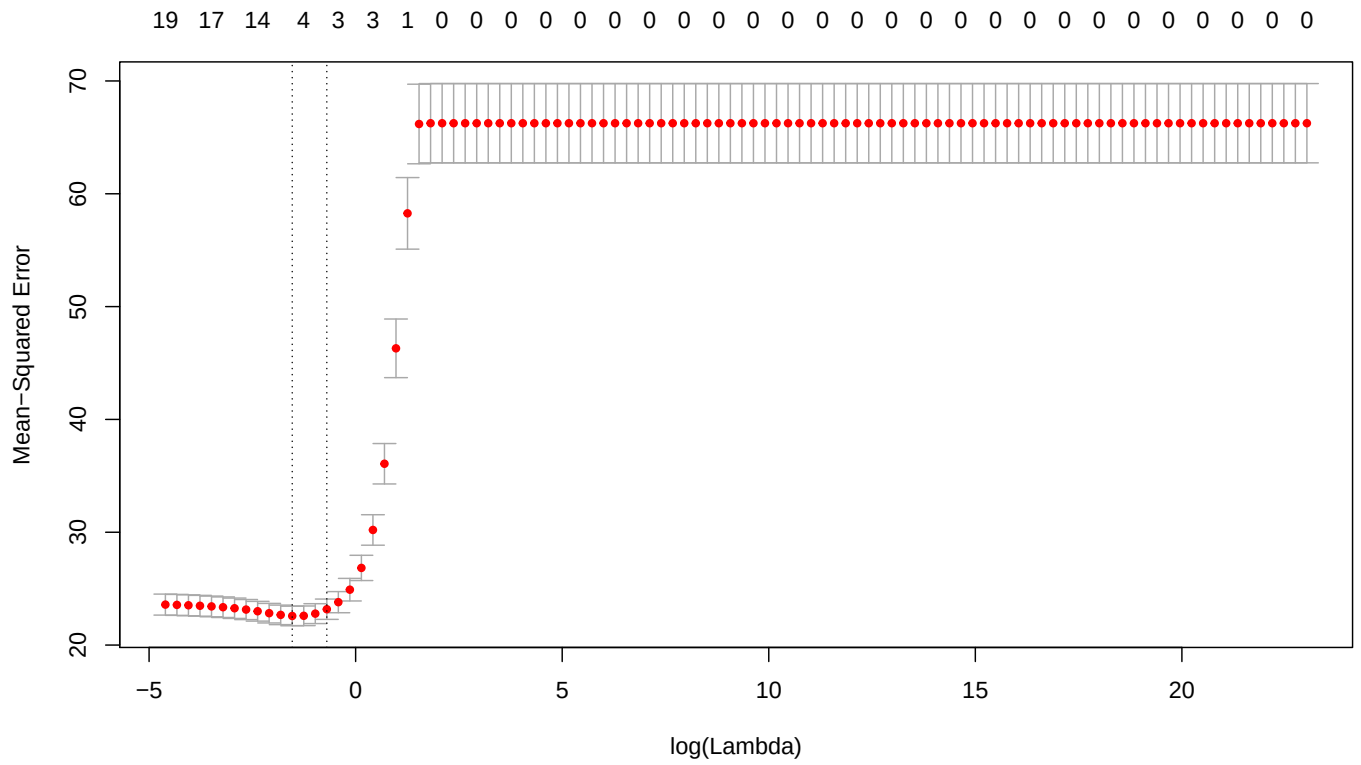
Lasso in R VI

```
> matplot(fitL$lambda[-(1:75)], t(coef(fitL)[-1,-(1:75)]), type="l")
```



Lasso in R - CV for lambda

```
> set.seed(4321)
> CVlasso <- cv.glmnet(x=X, y=y, alpha=1, lambda=lambda.grid)
> plot(CVlasso)
```



Lasso in R VI

```
> Bestlambda <- CVlasso$lambda.min
> Bestlambda   # minimizing OOS loss in CV
[1] 0.2154435
> Bestlambda2 <- CVlasso$lambda.1se
> Bestlambda2  # largest lambda 1 SE away from lambda.min
[1] 0.4977024
> coef(fitL)[-1, which(fitL$lambda == Bestlambda)]
        V1          V2          V3          V4          V5          V6
2.82870307  3.68828610  4.66026894  0.00000000  0.04429516  0.00000000
        V7          V8          V9         V10         V11         V12
0.00000000  0.00000000  0.00000000  0.00000000  0.00000000  0.00000000
       V13         V14         V15         V16         V17         V18
0.25319561  0.00000000  0.00000000  0.00000000  0.00000000  0.00000000
       V19         V20
0.00000000 -0.06096551
> coef(fitL)[-1, which(fitL$lambda == Bestlambda2)]
      V1       V2       V3       V4       V5       V6       V7       V8
2.532959 3.374796 4.358766 0.000000 0.000000 0.000000 0.000000 0.000000
      V9      V10      V11      V12      V13      V14      V15      V16
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
     V17      V18      V19      V20
0.000000 0.000000 0.000000 0.000000
```

Algorithms for the lasso

There are many algorithms for the lasso but in the following we give a simple one which is known as the **shooting algorithm** or **coordinate descent algorithm**.

The idea of this algorithm is that always one coefficient is estimated while the others are held fixed and the whole thing is iterated until some general convergence.

Coordinate Descent Algorithm

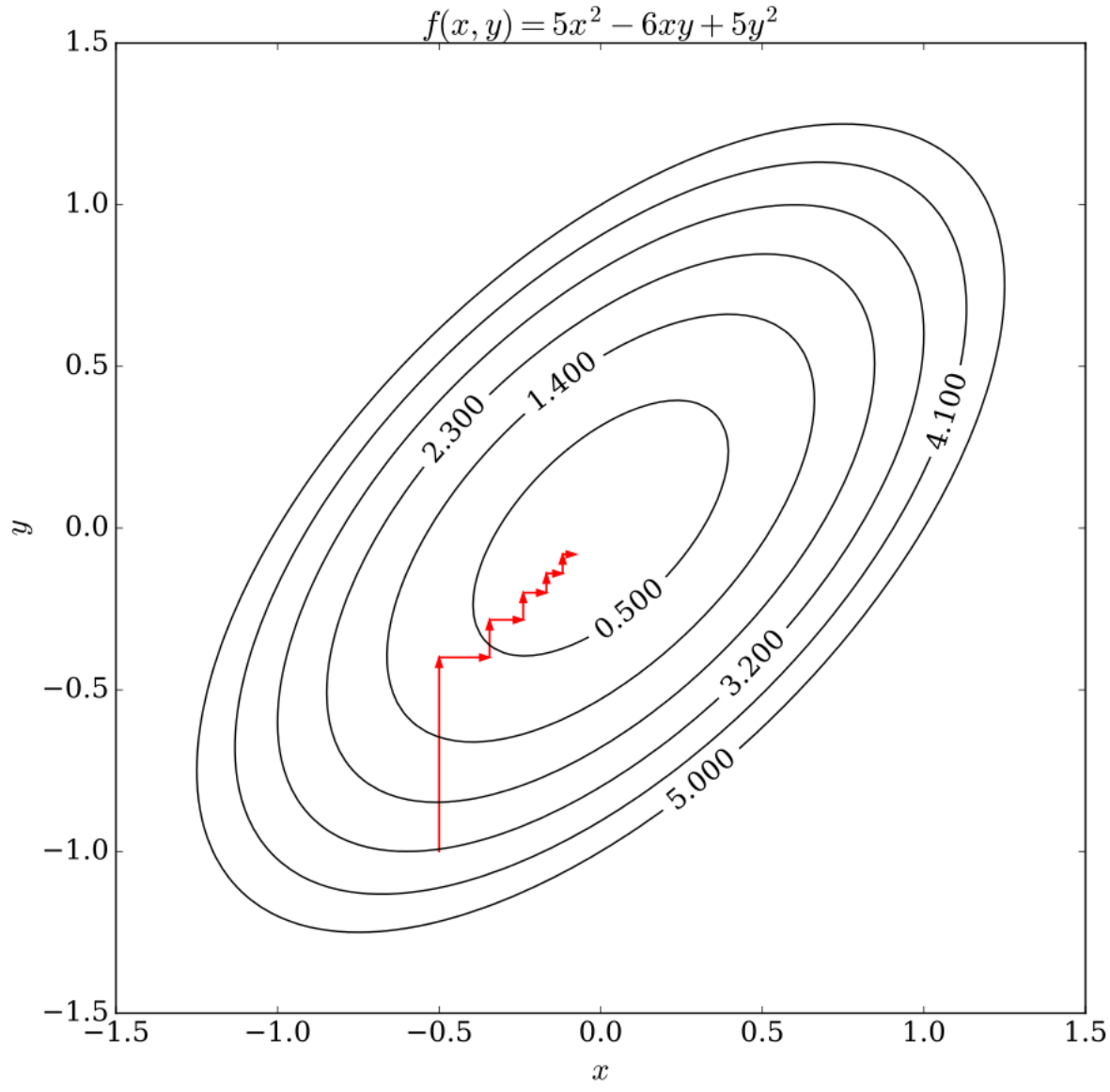
The algorithm iteratively solves the optimisation problem

$$x_i^{k+1} = \arg\min_{y \in \mathbb{R}} f(x_1^{k+1}, \ldots, x_{i-1}^{k+1}, y, x_{i+1}^{k}, \ldots, x_n^{k})$$

Using a line search algorithm leads to a sequence of x^i

$$F(\mathbf{x}^0) \geq F(\mathbf{x}^1) \geq F(\mathbf{x}^2) \geq \ldots$$

Coordinate Descent Algorithm



Shooting algorithm II

As we can obtain the non-regularized regression estimator $\hat{\beta}^0$ we can use this as a starting point for an algorithm to approximate the LASSO estimator.

Assume X is standardized data. Then a sequence of coefficient estimators $\beta^0, \beta^1, \beta^2, \ldots$ is constructed based on the following expressions for $j \in 1, \ldots, p$:

$$a_j = 2\sum_{i=1}^{n} x_{ij}^2 \quad \text{and } c_j = 2\sum_{i=1}^{n} x_{ij}(y_i - \boldsymbol{\beta}^{m\top}\mathbf{x}_i + \beta_j^m x_{ij})$$

Shooting algorithm

To update a_j and c_j we require the ancillary function `soft` which corresponds to so-called **soft thresholding**

$$soft(a, \delta) = sign(a) \cdot (|a| - \delta)_+,$$

where $x_+ = \max(x, 0)$ indicates the positive part of x.

Shooting algorithm II

Then, given a tolerance limit ϵ and a maximum number of iterations the algorithm repeats the following steps.

For $j \in 1, \ldots, p$ we use the above expressions

$$a_j = 2\sum_{i=1}^{n} x_{ij}^2 \quad \text{and } c_j = 2\sum_{i=1}^{n} x_{ij}(y_i - \boldsymbol{\beta}^{m\top}\mathbf{x}_i + \beta_j^m x_{ij})$$

and update the coefficients through soft thresholding

$$\beta_j^{m+1} = soft(c_j/a_j, \lambda/a_j)$$

The algorithm stops once $\sum_{j=1}^{p}(|\beta_j^{m+1} - \beta_j^m|) < \epsilon$ or the maximum number iterations is reached.

Problems with the lasso

Some problems with the lasso are:

- in high-dimensional datasets there are often highly correlated variables which means they all operate on the response in a similar way. In such a setup the lasso is known to perform not so well whereas ridge regression maybe a bit better.
- Often in high-dimensional datasets there are features which a structured together. In such a setting it would be natural to select either the whole group or drop it completely.

Generalizations of the Lasso

The ridge regression approach was a big step in high-dimensional regression but the lasso is the dominant method nowadays.

There are meanwhile many variants of the lasso improving various of the features of the regular lasso.

Some variants will be introduced in the following.

Elastic Net

So in this extreme case where `x1` and `x2` are almost identically correlated the unpenalized regression basically takes the sum of them with quite arbitrary values. The lasso arbitrarily assigns one coefficient all weight (first in model) and ridge regression gives both the same weight.

To address this issue the **elastic net** is a compromise between the ridge regression and lasso.

Elastic Net - Combining Ridge and LASSO

Consider a sample of size n, the elastic net coefficient vector β^{EN} minimizes

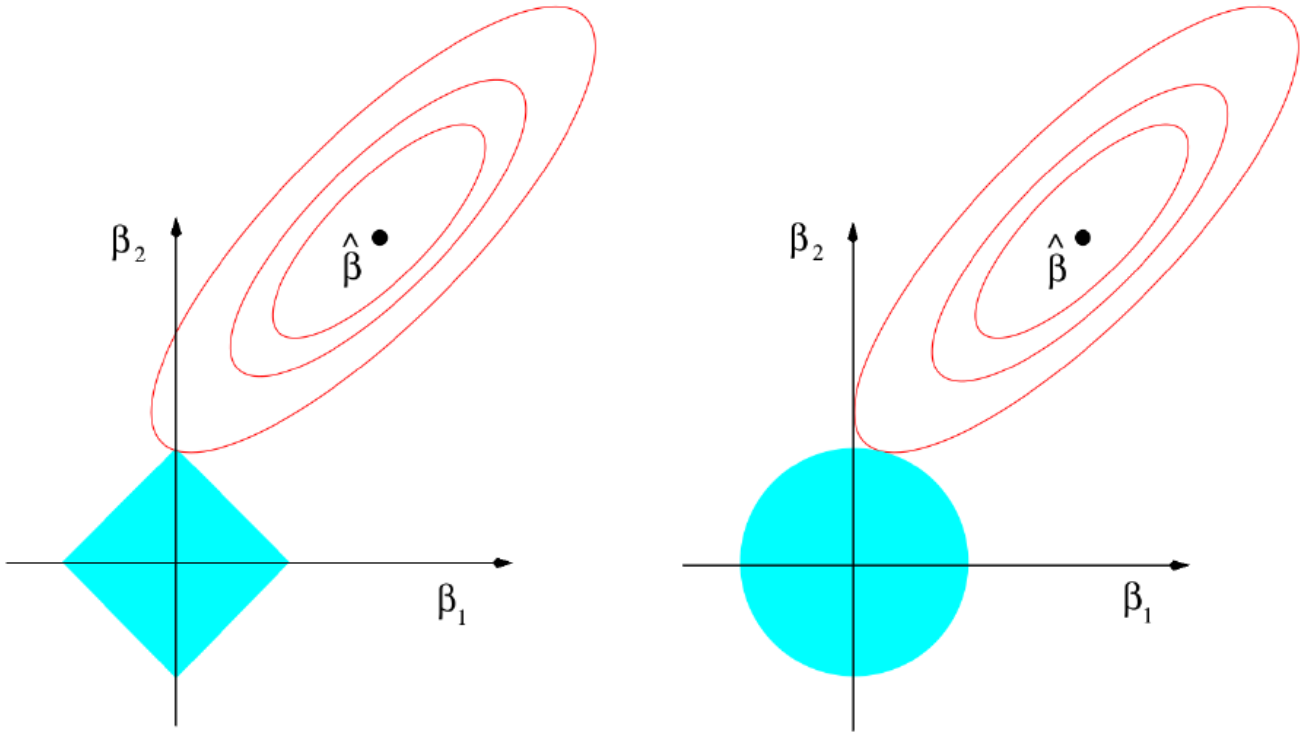
$$0.5\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)+\lambda\left[0.5(1-\alpha)\sum_{j=1}^{p}\beta_j^2+\alpha\sum_{j=1}^{p}|\beta_j|\right],$$

where α is a parameter than can be varied. Thus for a specific coefficient the penalty applied is

$$0.5(1-\alpha)\beta_j^2+\alpha|\beta_j|.$$

Thus for $\alpha = 1$ one obtains the lasso and for $\alpha = 0$ ridge regression.

Elastic net - penalties visualised



Elastic Net - Algorithmic background

As can be seen the elastic net estimator groups the correlated coefficients more natural together and still provides sparse solutions.

Some properties:

- the elastic net controls for strong within-group correlation.
- for $\alpha < 1$ and $\lambda > 0$ it is a strictly convex problem and therefore has a unique solution
- there are plenty of algorithms to solve the elastic net minimization problem. Coordinate descent is the most popular. The 0.5 for the quadratic part is mainly helping in the optimization.

Elastic Net - Choosing α

The question is then however how to choose the hyperparameter α?

Two common options:

1. Choose a value on subjective grounds and stick with that.
2. Choose a grid of αs and include it in a cross-validation procedure.

Elastic net in R

Consider the following extreme case

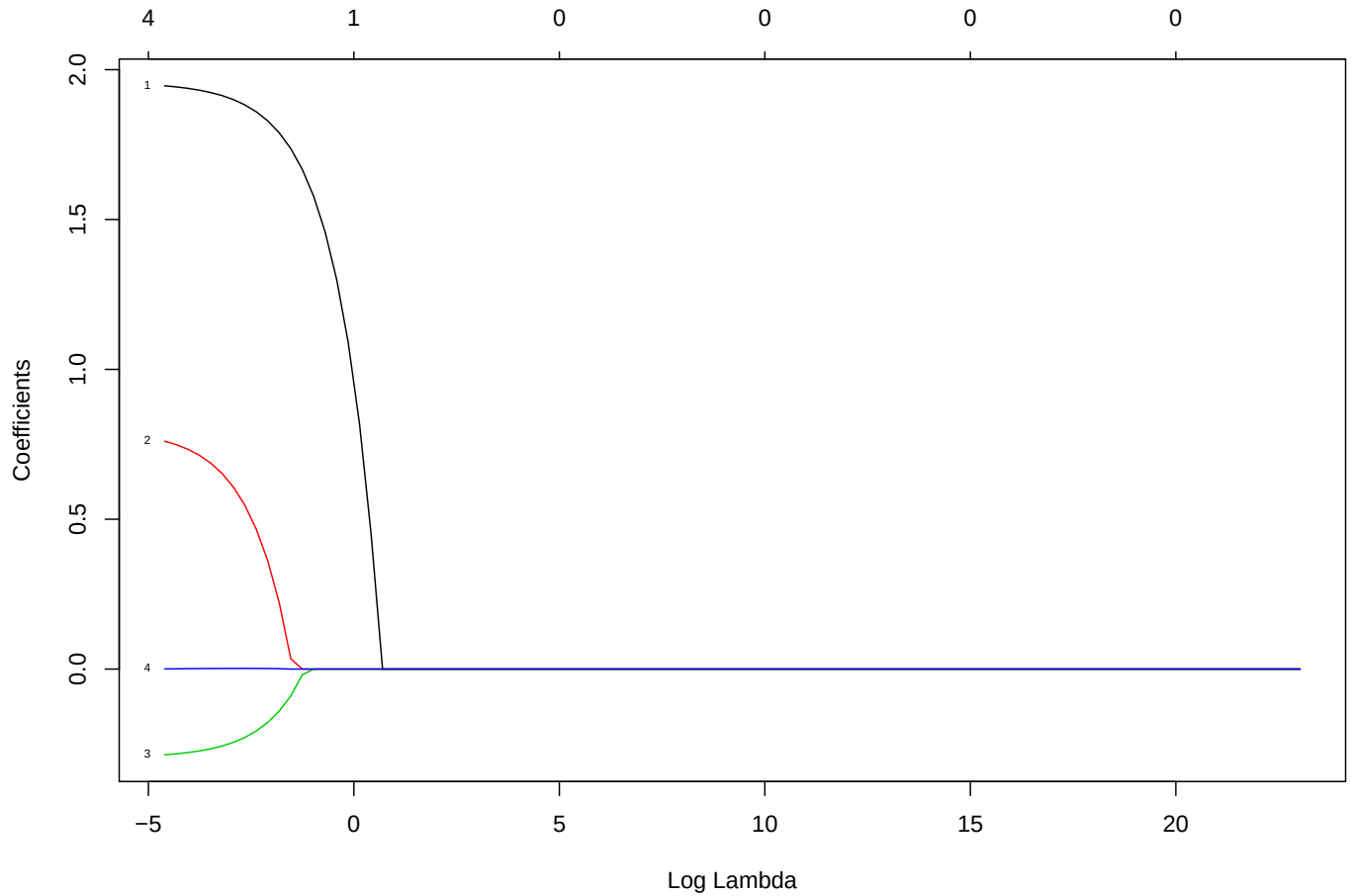
```
> library(glmnet)
> lambda.grid <- 10^seq(-2,10, length=100)
> set.seed(100)
> n <- 100
> x1 <- rnorm(n)
> x2 <- runif(n)
> x3 <- rnorm(n)
> eps <- rnorm(n, sd=2)
> x4 <- x1 + runif(n, -0.01, 0.01)
> X <- cbind(x1,x2,x3,x4)
> X2 <- cbind(x4,x2,x3,x1)
> cor(x1,x4)
[1] 0.9999824
>
> y <- 2 + 2*x1 + 1*x2 + eps
```

Elastic Net - Example Model

```
> summary(lm(y ~ x1 + x2 + x3 + x4))

Call:
lm(formula = y ~ x1 + x2 + x3 + x4)

Residuals:
    Min      1Q  Median      3Q     Max
-4.7271 -1.2059  0.2768  1.2480  4.8213

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   2.1322     0.4186   5.094 1.78e-06 ***
x1            6.7196    32.7365   0.205    0.838
x2            0.7961     0.7216   1.103    0.273
x3           -0.2933     0.2009  -1.460    0.148
x4           -4.7645    32.7504  -0.145    0.885
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.965 on 95 degrees of freedom
Multiple R-squared:  0.5249,    Adjusted R-squared:  0.5049
F-statistic: 26.24 on 4 and 95 DF,  p-value: 1.149e-14
```

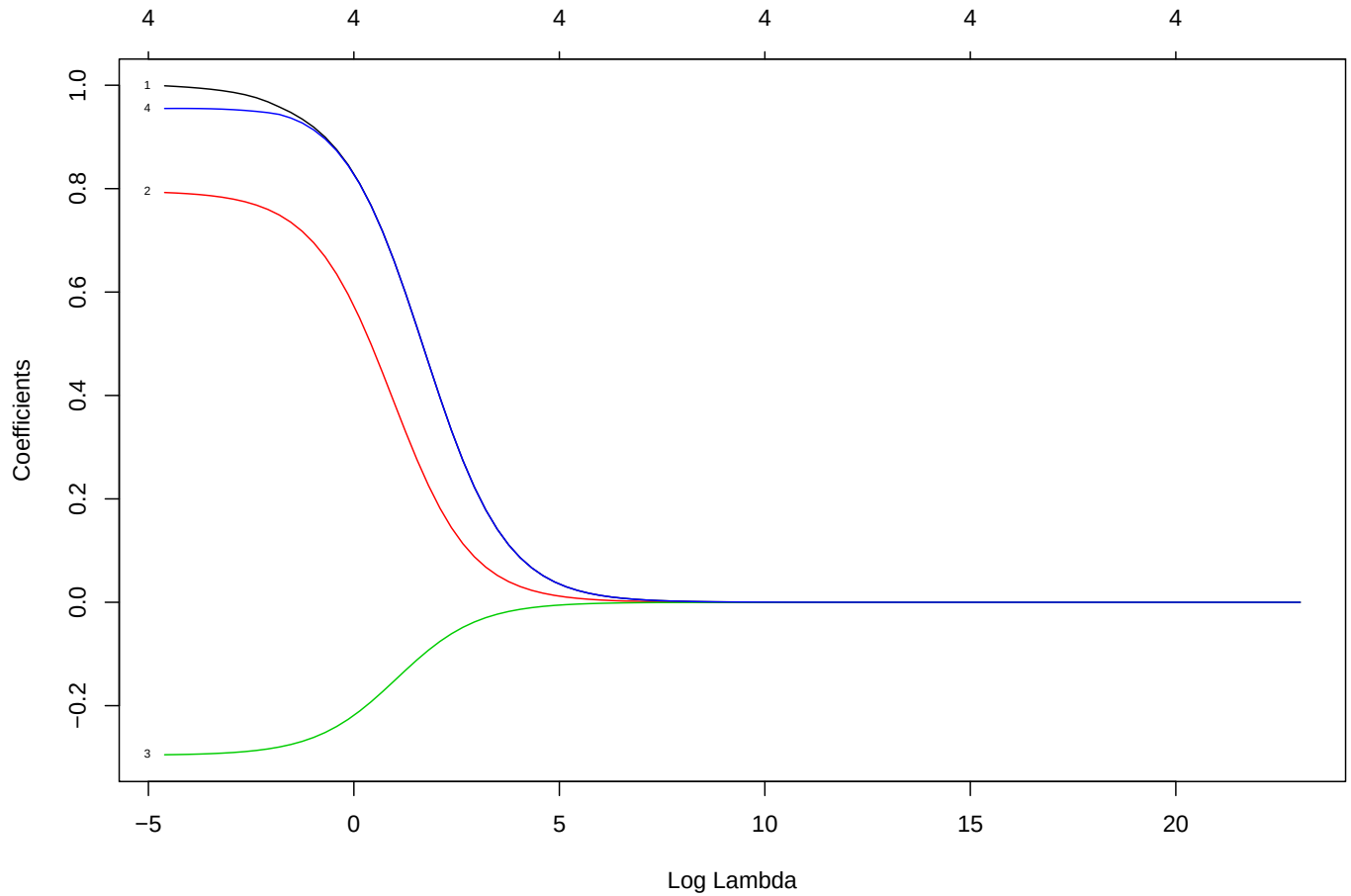
Elastic Net - LASSO extreme

```
> fitL <- glmnet(x=X, y=y, alpha=1, lambda=lambda.grid)
> plot(fitL,xvar="lambda",label=TRUE)
```



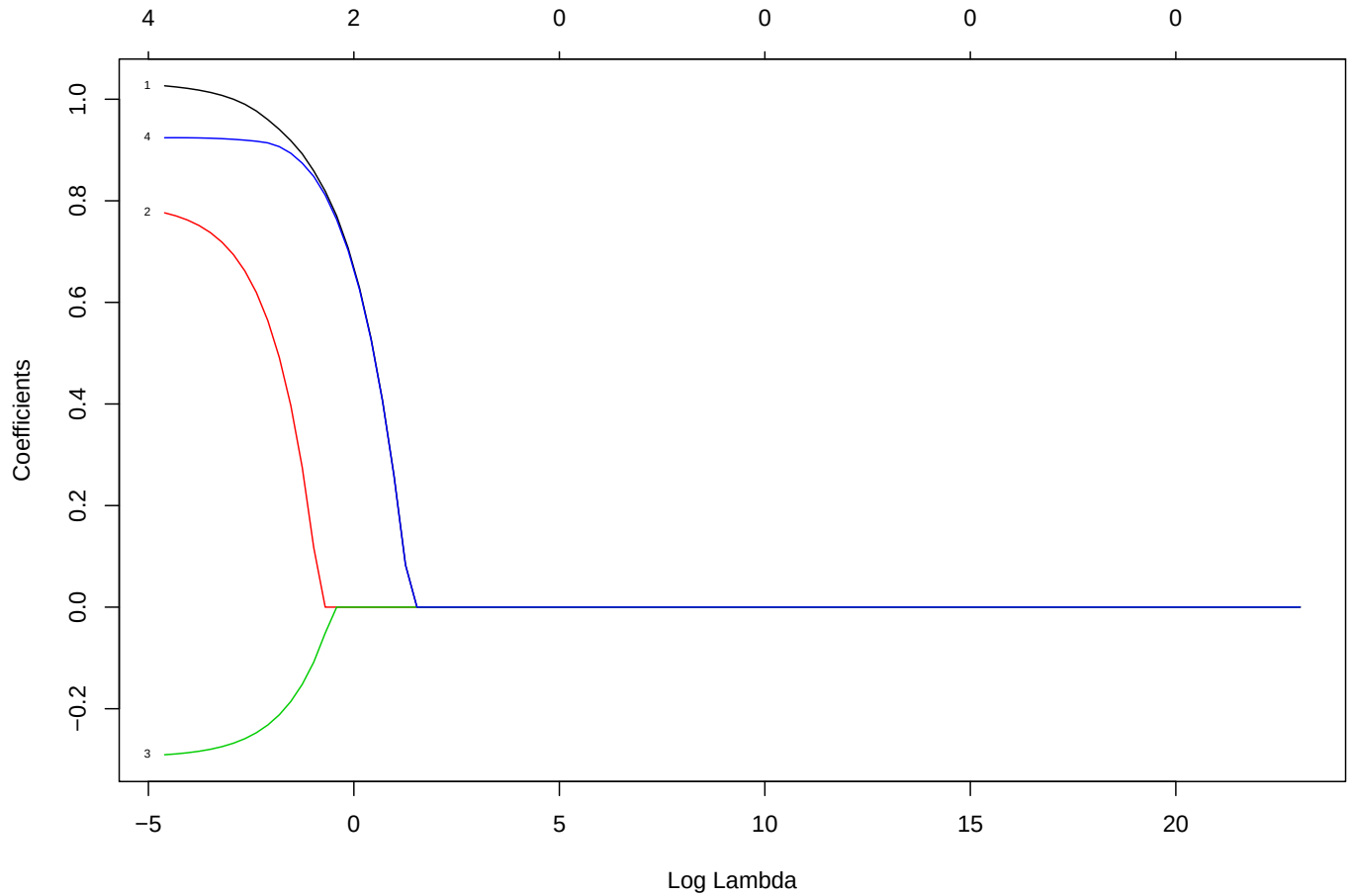
Elastic Net - Ridge extreme

```
> fitR <- glmnet(x=X, y=y, alpha=0, lambda=lambda.grid)
> plot(fitR,xvar="lambda",label=TRUE)
```



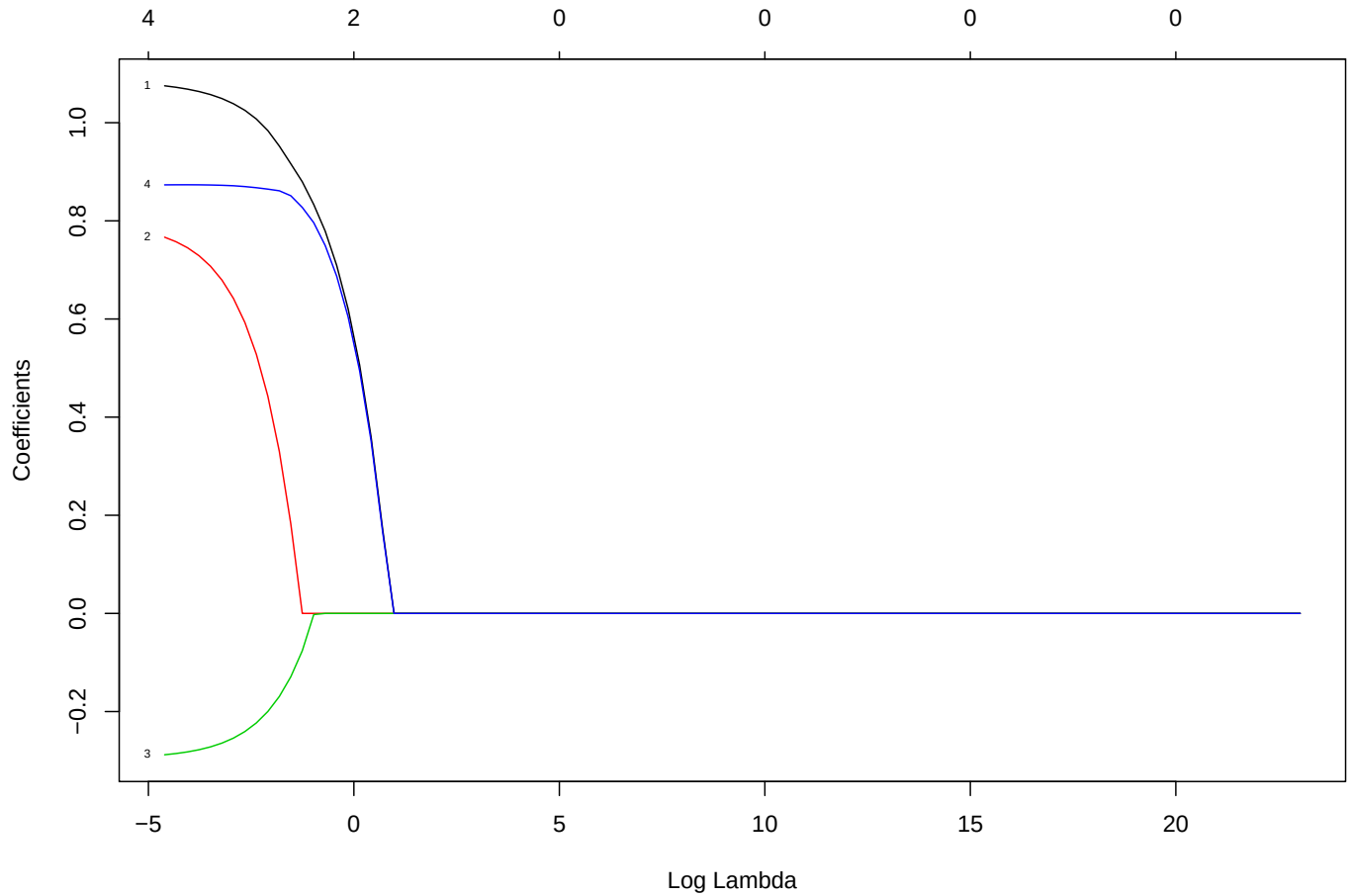
Elastic Net 50-50

```
> fitL2 <- glmnet(x=X, y=y, alpha=0.5, lambda=lambda.grid)
> plot(fitL2,xvar="lambda",label=TRUE)
```



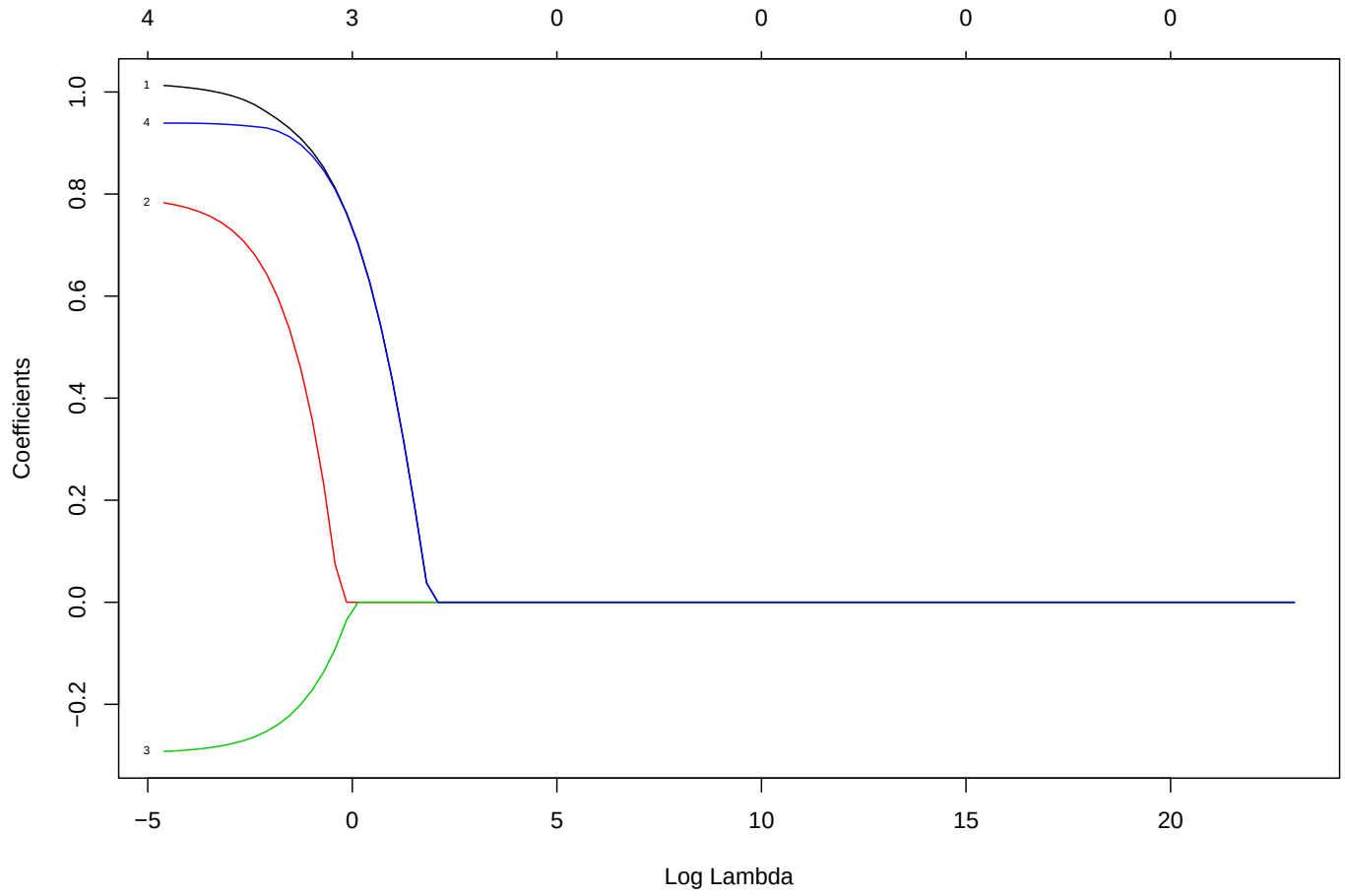
Elastic Net 80-20

```
> fitR2 <- glmnet(x=X, y=y, alpha=0.8, lambda=lambda.grid)
> plot(fitR2,xvar="lambda",label=TRUE)
```



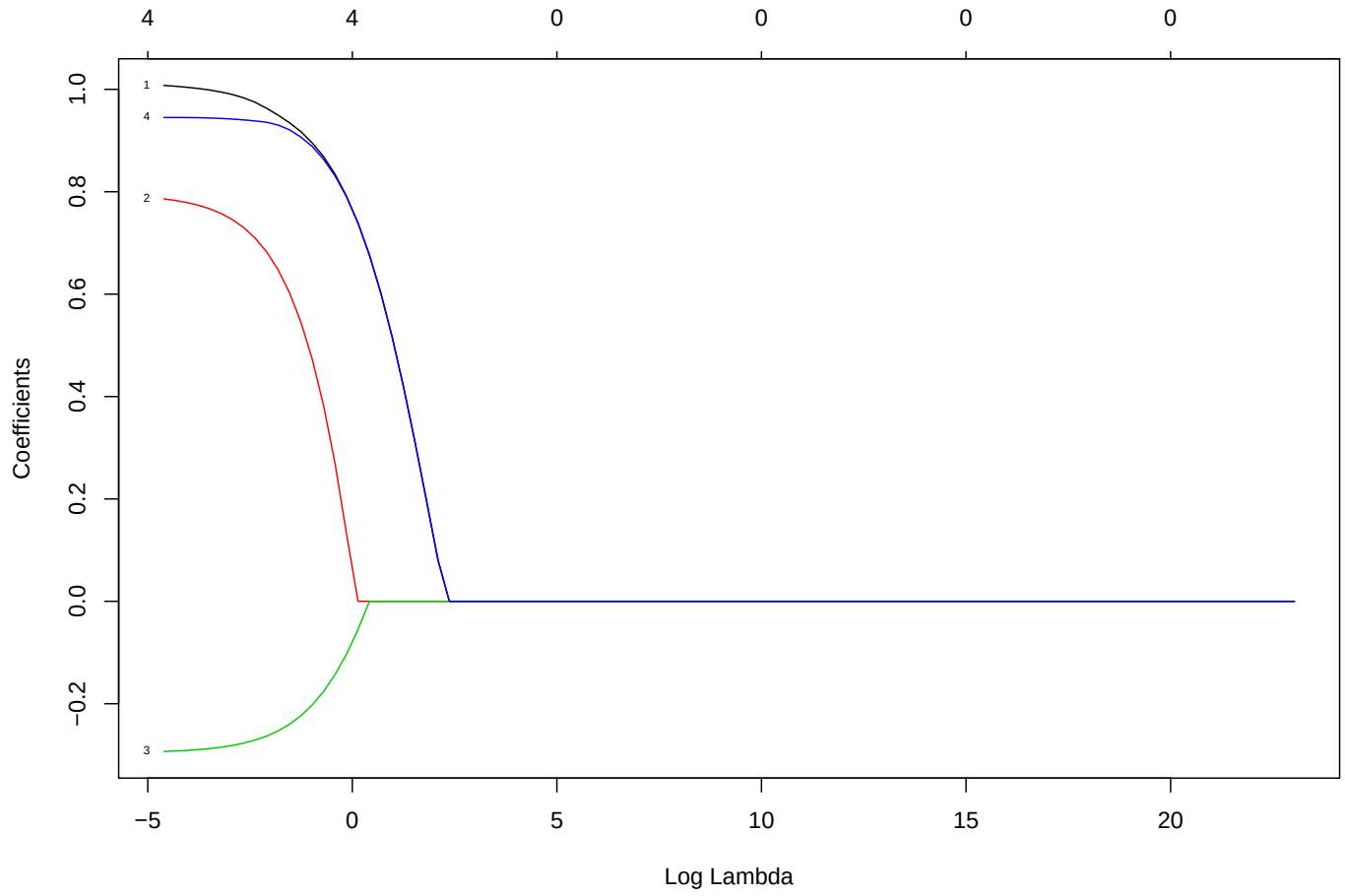
Elastic Net IX

```
> fitEN <- glmnet(x=X, y=y, alpha=0.3, lambda=lambda.grid)
> plot(fitEN,xvar="lambda",label=TRUE)
```



Elastic Net X

```
> fitEN2 <- glmnet(x=X, y=y, alpha=0.2, lambda=lambda.grid)
> plot(fitEN2,xvar="lambda",label=TRUE)
```



Group Lasso

There are many regression problems where a group of variables form a natural group structure and it would be desirable if all the coefficients in the group would be simultaneously nonzero or zero.

A common case is for a example a factor with L levels which is included in the model matrix via a contrast forming thus a group of size $L - 1$.

In the following we use the following notation:

- J denotes the number of groups of covariates
- for $j = 1, \ldots, J$ z_j consists of vector of the p_j covariates in group j.

Group Lasso II

Consider a sample of size n, the group lasso coefficient vector β^{GL} minimizes

$$0.5 \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{J} z_{ij}^{\top} \beta_j \right) + \lambda \sum_{j=1}^{J} ||\beta_j||_2,$$

where $|| \cdot ||_2$ denotes the Euclidean norm.

Group Lasso III

Properties of the group lasso:

- Depending on λ either the entire vector β_j will be zero or nonzero in a general setting.
- When all groups consist of single variables ($p_j = 1$ for all j) then $||\beta_j||_2 = |\beta_j|$ which means the problem reduces to the normal lasso.
- All groups are equally penalized, independent of the group size which makes larger groups more likely to be nonzero. There are formulations of the group lasso where the penalties of each group are weighted by group size to adjust for that.
- The computation of the group lasso is more sophisticated and done with either block coordinate descent or composite gradient methods.

Group Lasso IV

There are many R packages for the group lasso. We will use `gglasso`.

```
> library(gglasso)
> set.seed(1)
> n <- 100
> f1 <- gl(4,25)
> f2 <- gl(5,1,100)
> x1 <- rnorm(n)
> x2 <- rnorm(n)
> x3 <- rnorm(n)
> eps <- rnorm(n, sd=1.5)
> Beta <- c(10,0,0,0,1,1.5,2,2.5,2,0,0)
> X <- model.matrix(~f1+f2+x1+x2+x3)
> y <- X %*% Beta + eps
```

Group Lasso V

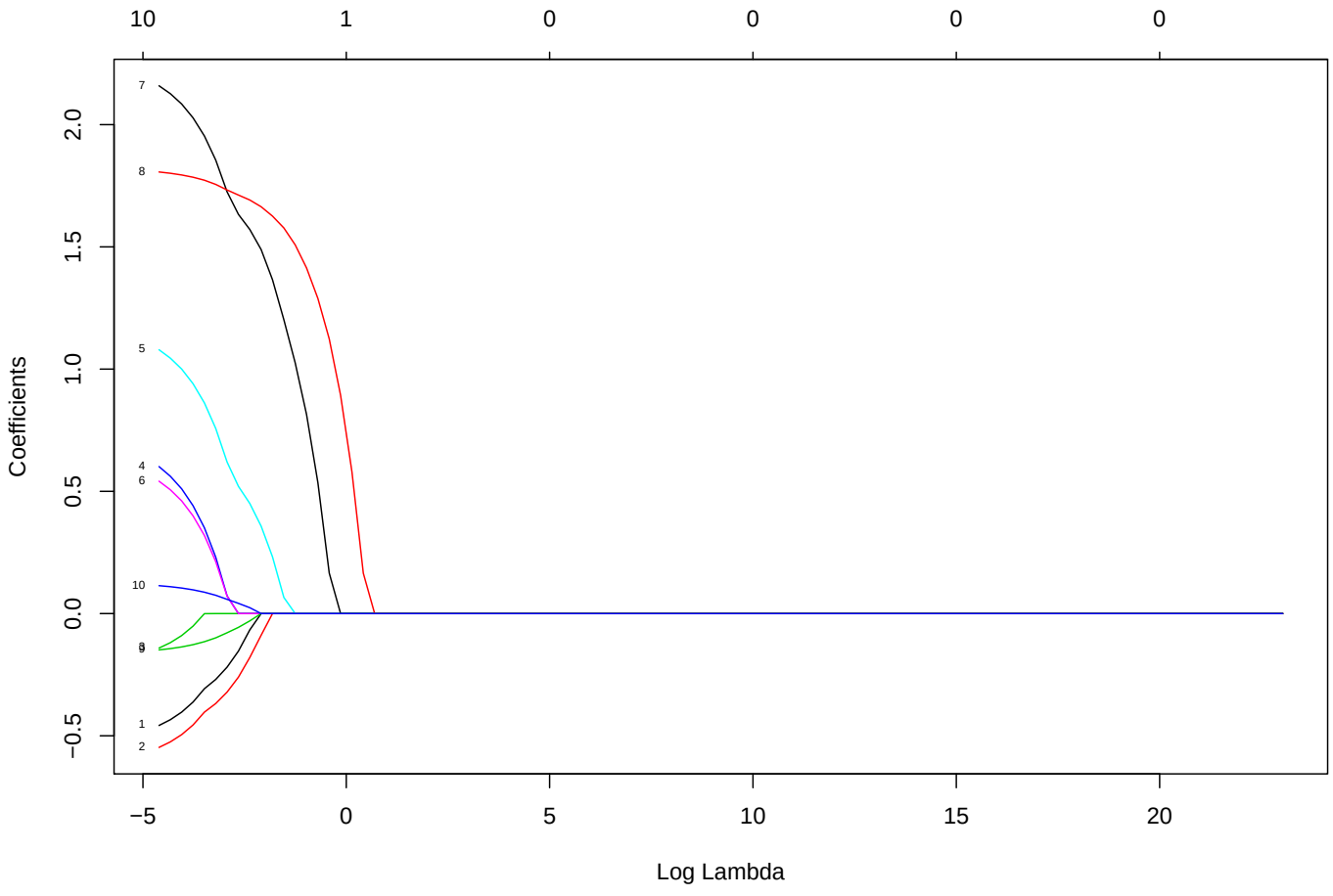
```
> head(X)
  (Intercept) f12 f13 f14 f22 f23 f24 f25         x1          x2         x3
1           1   0   0   0   0   0   0   0 -0.6264538 -0.62036668  0.4094018
2           1   0   0   0   1   0   0   0  0.1836433  0.04211587  1.6888733
3           1   0   0   0   0   1   0   0 -0.8356286 -0.91092165  1.5865884
4           1   0   0   0   0   0   1   0  1.5952808  0.15802877 -0.3309078
5           1   0   0   0   0   0   0   1  0.3295078 -0.65458464 -2.2852355
6           1   0   0   0   0   0   0   0 -0.8204684  1.76728727  2.4976616
> head(X[,-1])
  f12 f13 f14 f22 f23 f24 f25         x1          x2         x3
1   0   0   0   0   0   0   0 -0.6264538 -0.62036668  0.4094018
2   0   0   0   1   0   0   0  0.1836433  0.04211587  1.6888733
3   0   0   0   0   1   0   0 -0.8356286 -0.91092165  1.5865884
4   0   0   0   0   0   1   0  1.5952808  0.15802877 -0.3309078
5   0   0   0   0   0   0   1  0.3295078 -0.65458464 -2.2852355
6   0   0   0   0   0   0   0 -0.8204684  1.76728727  2.4976616
> GR <- c(1,1,1,2,2,2,2,3,4,5)
```

Group Lasso VI

```
> fitLS <- lm(y~ f1+f2+x1+x2+x3)
> summary(fitLS)

Call:
lm(formula = y ~ f1 + f2 + x1 + x2 + x3)

Residuals:
    Min      1Q  Median      3Q     Max
-3.9422 -0.7310  0.1741  0.8218  2.7323

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  10.8622     0.4180  25.983  < 2e-16 ***
f12          -0.5311     0.4215  -1.260    0.211
f13          -0.6179     0.4114  -1.502    0.137
f14          -0.2098     0.4118  -0.509    0.612
f22           0.7244     0.4783   1.514    0.133
f23           1.1861     0.4622   2.566    0.012 *
f24           0.6507     0.4692   1.387    0.169
f25           2.2598     0.4602   4.911 4.09e-06 ***
x1            1.8228     0.1672  10.905  < 2e-16 ***
x2           -0.1651     0.1633  -1.011    0.315
x3            0.1271     0.1440   0.882    0.380
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.454 on 89 degrees of freedom
```

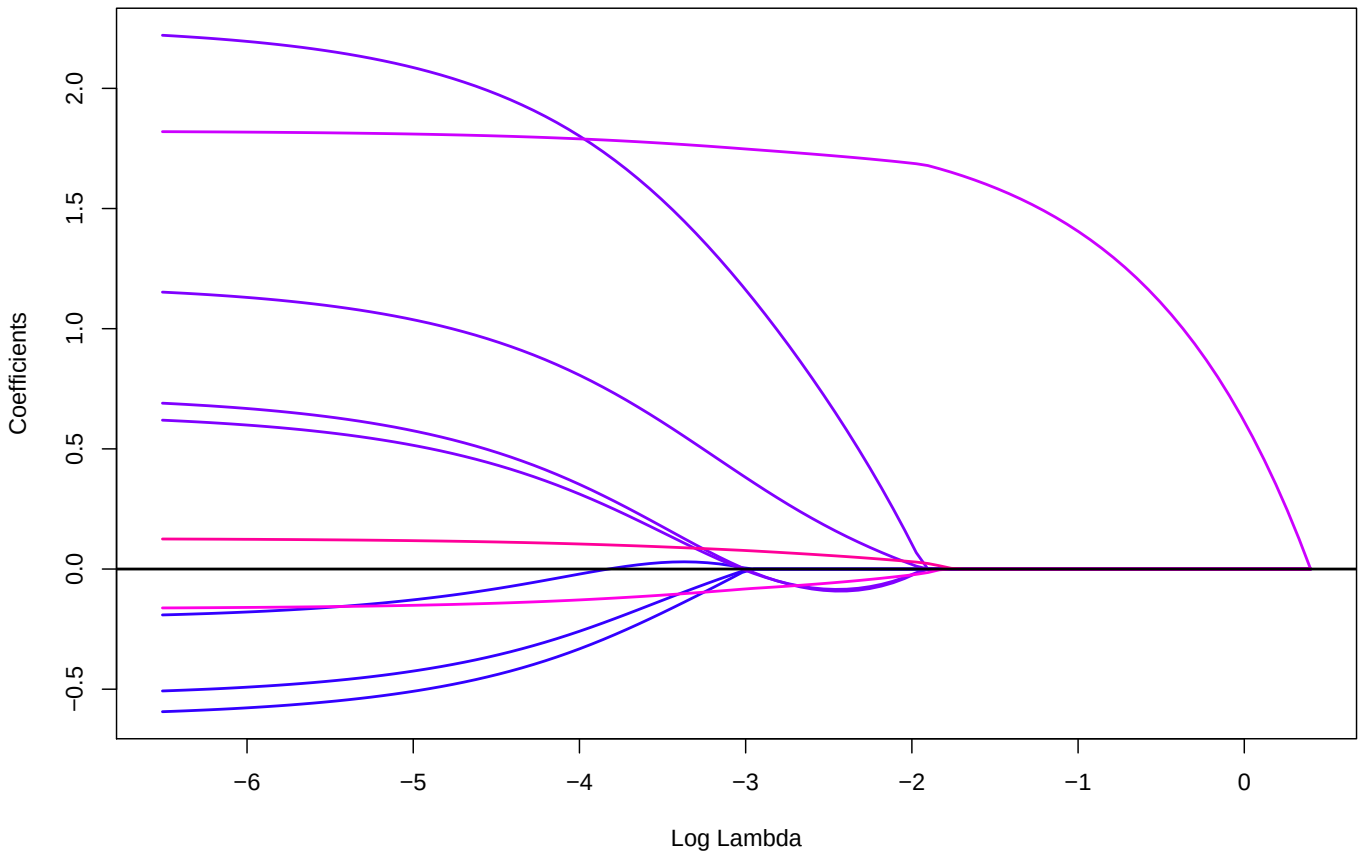
Group Lasso VII

```
> fitL <- glmnet(x=X[,-1], y=y, alpha=1, lambda=lambda.grid)
> plot(fitL, xvar="lambda", label=TRUE)
```



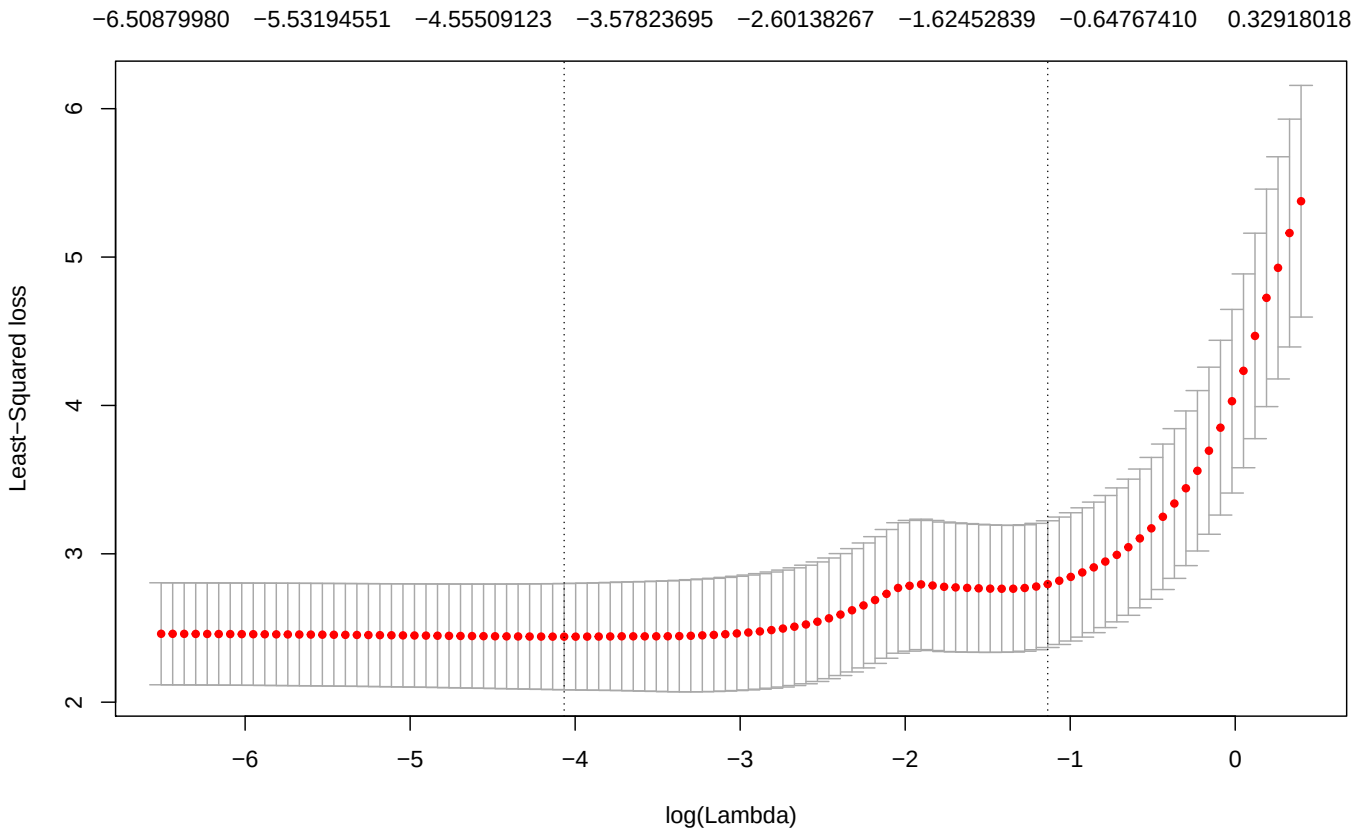
Group Lasso VIII

```
> fitGL <- gglasso(x=X[,-1], y=y, group=GR, loss="ls", intercept = TRUE)
> plot(fitGL)
```



Group Lasso IX

```
> CVfitGL <- cv.gglasso(x=X[,-1], y=y, group=GR, loss="ls",
+                       pred.loss="L2", nfolds=10)
> plot(CVfitGL)
```



Regularized multivariate methods

In general almost all multivariate methods have regularized variants nowadays.

And as in the regression case, there are usually many ways how to penalize features leading to different methods with different properties.

There are for example approaches to:

- sparse PCA
- sparse FA
- sparse CCA
- sparse LDA
- sparse clustering
- sparse graphical models
- sparse ...

and almost all of them have implementations available in R.

Exercises Task 1

1. Write your own function for the lasso using the shooting algorithm.
   - Choose default values for the tolerance limit and the maximum number of iterations. Do not forget to compute the coefficient for the intercept.
2. Write a function which computes the lasso using your algorithm for a vector of λ s and which returns the matrix of coefficients and the corresponding λ values.
3. Compare the performance and output of your functions against the lasso implementation from glmnet.
4. Write a function to perform 10-fold cross-validation for the lasso using MSE as the performance measure. The object should be similarly to the cv.glmnet give the same plot and return the λ which minimizes the Root Mean Squared Error, Mean Squared Error and Median Absolute Deviation, respectively.

Exercises - Task 2

We will work with the Hitters data in the ISLR package. Take the salary variable as the response variable and create the model matrix x based on all other variables in the data set. Then divide the data into training and testing data with a ratio of 70:30.

1. Use your lasso function to decide which lambda is best here. Plot also the whole path for the coefficients.
2. Compare your fit against the lasso implementation from glmnet.
3. Fit also a ridge regression and a least squares regression for the data (you can use here glmnet).
4. Compute the lasso, ridge regression and ls regression predictions for the testing data. Which method gives the better predictions? Interpret all three models and argue about their performances.

Exercises - Task 3

Explain the notion of regularised regression, shrinkage and how Ridge regression and LASSO regression differ.