Linear Regression

Given data
$$(y_i, \mathbf{x}_i) \quad i = 1, ..., n, \quad y_i \in R, \ \ \mathbf{x}_i \in R^p$$
we wish to find a real function $g(\mathbf{x})$ that optimally approximates the given data:
$$g(\mathbf{x}_i) \approx y_i, \qquad i- = 1, ..., n. \tag{1}$$
The function g may be assumed to belong to some space of functions, $\mathcal{G}$. For example $\mathcal{G}$ could be the set of all the linear functions
$$g(\mathbf{x}) = \beta_0 + \boldsymbol{\beta}'\mathbf{x},$$
leading to linear regression, or $\mathcal{G}$ could be the set of twice differentiable functions with bounded curvature leading to cubic splines. Moreover, the requirement that $g(\mathbf{x}_i)$ approximates y_i can be formalized in several ways, leading to different estimates $\widehat{g}(\mathbf{x}_i)$ including least squares (LS), least absolute regression, robust regression, etc.

Ordinary Least Squares Regression

$$\widehat{\boldsymbol{\theta}}_{LS} = \left(\widehat{\beta}_0, \widehat{\boldsymbol{\beta}}'\right) = \arg \min_{(\beta_0, \boldsymbol{\beta})} \sum \left(y_i - \beta_0 - \boldsymbol{\beta}'\mathbf{x}_i\right)^2$$
The matrix
$$X = \begin{pmatrix} 1 & x_{11} & \cdots & x_{1p} \\ 1 & x_{21} & \cdots & x_{2p} \\ \vdots & \vdots & & \vdots \\ 1 & x_{n1} & \cdots & x_{np} \end{pmatrix}$$
is called "the design matrix"and the vector
$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$
is called "the response vector". If X has full rank (that is, if $\text{rank}(X) = p+1 < n$) then it can be shown that
$$\widehat{\boldsymbol{\theta}} = (X'X)^{-1} X'\mathbf{y}.$$

Ridge Regression

One way to manage the bias-variance trade-off in linear regression is to add a penalty term, for example $\lambda \sum_{j=1}^{p} \beta_j^2$, to the loss function:
$$J(\beta_0, \boldsymbol{\beta}, \boldsymbol{\lambda}) = \sum_{i=1}^{n} \left(y_i - \beta_0 - \boldsymbol{\beta}'\mathbf{x}_i\right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \tag{2}$$

where $\lambda \geq 0$ is a tuning parameter. This technique was originally invented by the Russian mathematician Andrey Tikhonov and later expounded in statistics by Arthur E. Hoerl, who called it *ridge regression.*

Consider the problem of minimizing $J\left(\beta_0, \boldsymbol{\beta}, \boldsymbol{\lambda}\right)$ and let $\widehat{\boldsymbol{\beta}}(\lambda)$ be a minimizer of (2) for the given value of λ. If $\lambda = 0$, we are back to OLS, that is $\widehat{\boldsymbol{\beta}}(0) = \widehat{\boldsymbol{\beta}}_{OLS}$. On the other hand, it should be intuitively clear that if $\lambda \to \infty$, then $\widehat{\boldsymbol{\beta}}(\lambda) \to \mathbf{0}$. The effect of the penalty parameter λ is to shrink the regression coefficient toward zero.

Computation of $\widehat{\boldsymbol{\beta}}(\lambda)$.

In this context, it is convenient to center and to scale the variables so that

$$\sum_{i=1}^{n} y_i = \sum_{i=1}^{n} x_{ij} = 0,$$

to remove the need for the intercept parameter, β_0, and

$$\sum_{i=1}^{n} y_i^2 = \sum_{i=1}^{n} x_{ij}^2 = n$$

to account for the fact that $J\left(\boldsymbol{\beta},\lambda\right)$ is not scale invariant.

Notice that, for fixed λ,

$$J\left(\boldsymbol{\beta},\lambda\right) = \sum_{i=1}^{n} \left(y_i - \boldsymbol{\beta}'\mathbf{x}_i\right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

is differentiable and strictly convex, so if $\widehat{\boldsymbol{\beta}}(\lambda)$ solves the equation

$$\nabla J\left(\boldsymbol{\beta},\lambda\right) = -2 \begin{pmatrix} -\sum_{i=1}^{n} \left(y_i - \boldsymbol{\beta}'\mathbf{x}_i\right) x_{i1} + \lambda\beta_1 \\ -\sum_{i=1}^{n} \left(y_i - \boldsymbol{\beta}'\mathbf{x}_i\right) x_{i2} + \lambda\beta_2 \\ \vdots \\ -\sum_{i=1}^{n} \left(y_i - \boldsymbol{\beta}'\mathbf{x}_i\right) x_{ip} + \lambda\beta_p \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

then it $\widehat{\boldsymbol{\beta}}(\lambda)$ uniquely minimizes $J\left(\boldsymbol{\beta},\lambda\right)$ in $\boldsymbol{\beta}$ for this value of λ.

There are at least two approaches for calculating $\widehat{\boldsymbol{\beta}}(\lambda)$, which are described below.

Approach 1: (coordinate descent). Set $\boldsymbol{\beta}^0 = \mathbf{0}$ and given $\boldsymbol{\beta}^k = \left(\beta_1^k, \beta_2^k, ..., \beta_p^k\right)$

set

$$\begin{aligned}
e_{1i}^k &= y_i - \beta_2^k x_{i2} - \cdots - \beta_p^k x_{ip} \\
e_{2i}^k &= y_i - \beta_1^{k+1} x_{i1} - \beta_3^k x_{i3} - \cdots - \beta_p^k x_{ip} \\
e_{3i}^k &= y_i - \beta_1^{k+1} x_{i1} - \beta_2^{k+1} x_{i2} - \beta_4^k x_{i4} - \cdots - \beta_p^k x_{ip} \\
&\vdots \\
e_{pi}^k &= y_i - \beta_1^{k+1} x_{i1} - \beta_2^{k+1} x_{i2} - \cdots - \beta_4^{k+1} x_{i(p-1)}
\end{aligned}$$

We will use the convention: if $b < a$ then $\sum_{j'=a}^{b} d_{j'} = 0$ Notice that

$$\begin{aligned}
g(\beta_j) &= \sum_{i=1}^{n} \left(e_{ji}^k - \beta_j x_{ji}\right)^2 + \lambda \beta_j^2 + \lambda \left(\sum_{j'=1}^{j-1} \left(\beta_{j'}^{k+1}\right)^2 + \sum_{j'=j+1}^{p} \left(\beta_{j'}^k\right)^2 \right) \\
&= \sum_{i=1}^{n} \left(e_{ji}^k - \beta_j x_{ji}\right)^2 + \lambda \beta_j^2 + C
\end{aligned}$$

is a convex and diferentiable function of β_j. So the solution of

$$-2 \sum_{i=1}^{n} \left(e_{ji}^k - \beta_j x_{ji}\right) x_{ji} + 2\lambda \beta_j = 0$$

is the unique minimizer of $g(\beta_j)$. We have

$$\sum_{i=1}^{n} \left(e_{ji}^k - \beta_j x_{ji}\right) x_{ji} + \lambda \beta_j = -\sum_{i=1}^{n} e_{ji}^k x_{ji} + \beta_j (\lambda + 1) = 0$$

$$\beta_j^{k+1} = \frac{\sum_{i=1}^{n} e_{ji}^k x_{ji}}{1 + \lambda}, \quad \text{for} \quad j = 1, 2, ..., p$$

Approach 2: (expanded OLS): We can view (2) as an expanded OLS problem with expanded response vector

$$J(\boldsymbol{\beta}, \boldsymbol{\lambda}) = \left\| \widetilde{\mathbf{y}} - \widetilde{X} \boldsymbol{\beta} \right\|^2$$

with

$$\widetilde{\mathbf{y}}_{(n+p)\times 1} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \begin{pmatrix} \mathbf{y}_{n\times 1} \\ \mathbf{0}_{p\times 1} \end{pmatrix} \quad \text{and} \quad \widetilde{X} = \begin{pmatrix} X_{n\times p} \\ \sqrt{\lambda} I_p \end{pmatrix}.$$

Then

$$\boldsymbol{\beta}(\lambda) = \left(\widetilde{X}' \widetilde{X}\right)^{-1} \widetilde{X}' \widetilde{\mathbf{y}}$$

Cross Validation

All the formulas and derivations so far were made for a fixed value of λ. Naturally the results and conclusions of any data analysis will depend on the particular value of λ. Therefore, the problem of estimating λ is of much practical importance.

There is no point in minimizing $J(\boldsymbol{\beta}(\lambda), \lambda)$ because the best fit to the training data is obviously obtained at $\lambda = 0$, the unconstrained minimizer. However, in some situations assigning a positive value to λ may be convenient to prevent *overfitting of the training data.* This is another manifestation of the bias-variance trade-off discussed at the beginning of this section.

To choose λ we search for a model with a good out-of-sample generalization performance. In principle, to optimize the out-of-sample performance we need an independent test dataset $\{(y_i, \mathbf{x}_i)\}_{i=n+1}^{n+m}$ to compute

$$\widehat{\lambda} = \arg\min \sum_{i=n+1}^{n+m} \left(y_i - \mathbf{x}_i' \widehat{\boldsymbol{\beta}}(\lambda)\right)^2 .$$

However, in most situations, the test dataset $\{(y_i, \mathbf{x}_i)\}_{i=n+1}^{n+m}$ is not available. A practical solution consists of splitting the training dataset $\{(y_i, \mathbf{x}_i)\}_{i=1}^{n}$ in two parts, one used to estimate $\widehat{\beta}(\lambda)$ (for fixed values of λ) and the other to estimate λ. Naturally, different splits of the data may lead to different estimates $\widehat{\lambda}$.

At this point we need to set some notations. Let

$$I = \{1, 2, ..., n\},$$

and consider random partitions of size n_1 :

$$I_1 \cup I_2 = I, \quad I_1 \cap I_2 = \phi, \quad \# I_1 = n_1,$$

for some fixed number $1 \leq n_1 \leq n$. Now we describe the following algorithm.

1. Let $\mathcal{D} = \{0 = \lambda_1 < \lambda_2 < \cdots < \lambda_K\}$, an appropriate grid of values of λ.
2. Let $\left\{\left(I_1^b, I_2^b\right) : \# I_1^b = n_1, b = 1, ..., B\right\}$
3. For each pair (b, k), $b = 1, ..., B$ and $k = 1, ..., K$ compute

$$\boldsymbol{\beta}^b(\lambda_k) = \arg\min_{\boldsymbol{\beta}} \left[\sum_{i \in I_1^b} \left(y_i - \mathbf{x}_i' \boldsymbol{\beta}\right)^2 + \lambda_k \|\boldsymbol{\beta}\|^2\right]$$

$$G(\lambda_k, b) = \sum_{i \in I_2^b} \left[y_i - \mathbf{x}_i' \boldsymbol{\beta}^b(\lambda_k)\right]^2$$

4. Form the estimated generalization error for each λ_k, $k = 1, ..., K$

$$G(\lambda_k) = \frac{1}{B}\sum_{b=1}^{B} G(\lambda_k, b)$$

5. Set

$$\widehat{\lambda} = \arg \min_{1 \le k \le K} G(\lambda_k).$$

6. Output $\widehat{\lambda}$ and

$$\widehat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \left[\sum_{i=1}^{n} (y_i - \mathbf{x}_i'\boldsymbol{\beta})^2 + \widehat{\lambda} \|\boldsymbol{\beta}\|^2 \right]$$

Leave-One-Out Crossvalidation:

The particular case where $n_1 = n - 1$ leads to the popular approach called *leave-one-out crossvalidation.* In this case instead of random splits we consider all the possible n splits with the j^{th} case set aside for testing and the remaining cases used for training.

L_1-Regression

We will see that OLS is non-resistent to the presence of outliers in the training data. The reason for the lack of robustness is the rapid increase of the quadratic loss function . Some resistance can be gained by replacing the quadratic loss function $\rho(x) = x^2$ by the L_1-loss function$\rho(x) = |x|$. In this case,

$$\widehat{\boldsymbol{\theta}}_{L_1} = \left(\widehat{\beta}_0, \widehat{\boldsymbol{\beta}}\right) = \arg \min_{(\beta_0, \boldsymbol{\beta})} \sum \left| y_i - \beta_0 - \boldsymbol{\beta}' \mathbf{x}_i \right|. \tag{3}$$

The corresponding L_1-regression loss function,

$$J(\beta_0, \boldsymbol{\beta}) = \sum \left| y_i - \beta_0 - \boldsymbol{\beta}' \mathbf{x}_i \right|,$$

is convex but not differentiable. In this case, computing a global minimizer, $\left(\widehat{\beta}_0, \widehat{\boldsymbol{\beta}}\right)$, is considerable harder, compared with the OLS case.To better deal with (3) and for other derivations we can use the concept of *subgradient of a convex function.*

First consider the simpler case when $p = 1$. that is, consider the loss function:

$$J(\beta_0, \beta_1) = \sum_{i=1}^{n} |y_i - \beta_0 - \beta_1 x_i|$$

We will use the method of coordinate descent discussed before (see Theorem 1).

First, set $\beta_1 = 0$ and minimize

$$J(\beta_0, 0) = \sum_{i=1}^{n} |y_i - \beta_0| . \tag{4}$$

The solution to (4) is

$$\widehat{\beta}_0^0 = \mathrm{Med}(y_i)$$

Second, set $\beta_0 = \widehat{\beta}_0^0$ and minimize

$$g_0(\beta_1) = J\left(\widehat{\beta}_0^0, \beta_1\right) = \sum_{i=1}^{n} \left| \left(y_i - \widehat{\beta}_0^0\right) - \beta_1 x_i \right| = \sum_{i=1}^{n} |\widetilde{y}_i - \beta_1 x_i| \tag{5}$$

with

$$\widetilde{y}_i = y_i - \widehat{\beta}_0^0.$$

Now

$$\partial g_0(\beta_1) = \sum_{i=1}^{n} \partial |\widetilde{y}_i - \beta_1 x_i| = \sum_{\{i:x_i \neq 0\}} \partial |\widetilde{y}_i - \beta_1 x_i|$$

$$= \sum_{\{i:x_i \neq 0\}} |x_i| \, \partial \left| \frac{\widetilde{y}_i}{x_i} - \beta_1 \right| = 0$$

Equivalently, we wish to solve

$$\sum_{\{i:x_i \neq 0\}} w_i \partial |z_i - \beta_1| = 0$$

with

$$z_i = \frac{\widetilde{y}_i}{x_i} \quad \text{and} \quad w_i = \frac{|x_i|}{\sum_{\{j:x_j \neq 0\}} |x_j|}.$$

At this point, for simplicity, we will assume that $z_1 < z_2 < \cdots < z_m$, with $m = \#\{j : |x_j| > 0\}$. If this is not the case, we must work with the distinct values of the z_i's and associate each of them with the sum of the corresponding $|x_j|'$s. Consider the following distribution

i	z	w	F
1	z_1	w_1	$F_1 = w_1$
2	z_2	w_2	$F_2 = w_1 + w_2$
3	z_3	w_3	$F_3 = w_1 + w_2 + w_3$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
m	z_m	w_m	$F_m = 1$

At this point we consider two cases:

Case 1. There exist $1 \leq k < m$ such that $F_k = 0.5$
Let $z_k < \beta_1 < z_{k+1}$ and notice that in this case

$$\sum_{i=1}^{m} w_i \partial \left| z_i - \beta_1 \right| = \sum_{i=1}^{k} w_i - \sum_{i=k+1}^{m} w_i = 0.$$

Therefore, for example

$$\widehat{\beta}_1^1 = \frac{z_k + z_{k+1}}{2}$$

minimizes $g\left(\beta_1\right)$ given by (5).

Case 2. There exist $1 \leq k < m$ such that $F_k < 0.5$ and $F_{k+1} > 0.5$
Let $\beta_1 = z_k$ and notice that in this case, for all $-1 \leq \delta \leq 1$,

$$\sum_{i=1}^{k-1} w_i + \delta w_k - \sum_{i=k+1}^{m} w_i \in \sum_{i=1}^{m} w_i \partial \left| z_i - \beta_1 \right|$$

$$\delta = \frac{\sum_{i=k+1}^{m} w_i - \sum_{i=1}^{k-1} w_i}{w_k}$$

(which is between -1 and 1, why?) we conclude that $\widehat{\beta}_1^1 = z_k$ minimizes $g\left(\beta_1\right)$ given by (5). In summary, let

$$k = \max\left\{j : F_j \leq 0.5\right\}$$

If $F_k = 0.5$, take

$$\widehat{\beta}_1^1 = \frac{z_k + z_{k+1}}{2}$$

otherwise take

$$\widehat{\beta}_1^1 = z_k.$$

In fact, in all cases, $\widehat{\beta}_1^1 = z_k$ always minimizes $g\left(\beta_1\right)$ given by (5) (why?).

Computing Algorithm

Based on the discussion above we apply the following iterative algorithm. Set

$$s = \sum_{j=1}^{n} \left| x_j \right|$$

1. **Input.** The values $\left(x_i, y_i\right), \quad i = 1, ..., n$
2. **Initialization.** Set $\boldsymbol{\theta}^0 = \left(\beta_0^0, \beta_1^0\right) = \left(Med\left(y_i\right), 0\right)$ and choose a value for setting the absolute precision δ.

3. **Iteration.** While

$$\left|\beta_0^{k+1} - \beta_0^k\right| > \left|\beta_0^k\right| \delta \quad \text{and} \quad \left|\beta_1^{k+1} - \beta_1^k\right| > \left|\beta_1^k\right| \delta,$$

given $\boldsymbol{\theta}^k = \left(\beta_0^k, \beta_1^k\right)$,compute $\boldsymbol{\theta}^{k+1} = \left(\beta_0^{k+1}, \beta_1^{k+1}\right)$ as follows:

(a) Ignore for this calculation cases with $x_i = 0$.

(b) Set, for $x_i \neq 0$,

$$z_i = \frac{y_i - \beta_0^k}{x_i} \quad , \quad w_i = \frac{|x_i|}{s}$$

(c) Denote by ζ_j $(j = 1, ..., m)$ the sorted, distinct values of z_i, and denote by π_j their corresponding weights. For example, if z_{i_1} and z_{i_2} are the only two values of z_i equal to ζ_1 then $\pi_1 = z_{i_1} + z_{i_2}$.

(d) Calculate

$$F_j = \sum_{l=1}^{j} \pi_l, \quad l = 1, ..., m$$

(e) Set

$$j^* = \max\left\{j : F_j \leq 0.5\right\}$$

If $F_{j^*} = 0.5$, set

$$\beta_1^{k+1} = \frac{\zeta_{j^*} + z_{j^*+1}}{2}$$

If $F_{j^*} < 0.5$ set

$$\beta_1^{k+1} = \zeta_{j^*}$$

4. Set

$$\beta_0^{k+1} = \operatorname*{Med}_{i=1,\ldots,n} \left\{y_i - x_i \beta_1^{k+1}\right\}$$

5. **Output:** After stop, return the values $\left(\beta_0^{k+1}, \beta_1^{k+1}\right)$.

Note: If we have $p > 1$, steps a-e are modified as follows. First, instead of x_i we have x_{qi}, $q = 1, ..., p$ and instead of s we have

$$s_q = \sum_{j=1}^{n} |x_{qi}|.$$

Moreover, instead of

$$z_i = \frac{y_i - \beta_0^k}{x_i} \quad , \quad w_i = \frac{|x_i|}{s}$$

we have

$$z_{qi} = \frac{y_i - \beta_0^k - \sum_{l \neq q} \beta_l^k x_{li}}{x_{qi}} \quad , \quad w_{qi} = \frac{|x_{qi}|}{s_q}.$$

The remaining calculations are modified in an obvious way.

LASSO

Another way to manage the bias-variance trade-off in linear regression is to add the penalty term $\lambda \sum_{j=1}^{p} |\beta_j|$, to the quadratic loss function:

$$J\left(\beta_0, \boldsymbol{\beta}, \boldsymbol{\lambda}\right) = \frac{1}{2}\sum_{i=1}^{n}\left(y_i - \beta_0 - \boldsymbol{\beta}'\mathbf{x}_i\right)^2 + \lambda\sum_{j=1}^{p}|\beta_j| \tag{6}$$

As in the case of ridge regression, it is convenient to center and scale the variables so that

$$\sum_{i=1}^{n} y_i = \sum_{i=1}^{n} x_{ij} = 0,$$

and

$$\sum_{i=1}^{n} y_i^2 = \sum_{i=1}^{n} x_{ij}^2 = 1$$

With this pre-processing (6) become

$$J\left(\boldsymbol{\beta}, \boldsymbol{\lambda}\right) = \frac{1}{2}\sum_{i=1}^{n}\left(y_i - \boldsymbol{\beta}'\mathbf{x}_i\right)^2 + \lambda\sum_{j=1}^{p}|\beta_j|$$

Notice that, for fixed λ, $J\left(\boldsymbol{\beta}, \boldsymbol{\lambda}\right)$ is sub-differentiable and convex, and so it has a global minimizer $\widehat{\boldsymbol{\beta}}(\lambda)$. In order to implement a coordinate-descent algorithm to compute $\widehat{\boldsymbol{\beta}}(\lambda)$, we will derive a "close form formula" for the case $p = 1$. That is, we consider the minimization problem

$$J\left(\beta, \boldsymbol{\lambda}\right) = \frac{1}{2}\sum_{i=1}^{n}\left(y_i - \beta x_i\right)^2 + \lambda|\beta|$$

and the first order condition

$$0 \in \partial J\left(\beta, \boldsymbol{\lambda}\right) = \left\{-\sum_{i=1}^{n}\left(y_i - \beta x_i\right)x_i + \lambda\partial|\beta|\right\}$$

or equivalently

$$0 \in \left\{\sum_{i=1}^{n} x_i y_i - \beta - \lambda\partial|\beta|\right\}$$

which is in turn equivalent to

$$\left(\sum_{i=1}^{n} x_i y_i - \beta\right) \in \lambda\partial|\beta| \tag{7}$$

We set $r = \sum_{i=1}^{n} x_i y_i$ and consider two cases.

Case 1: $|r| > \lambda$

If $r > \lambda$, then setting $\beta(\lambda) = r - \lambda$ solves (7) because $\beta(\lambda) = r - \lambda > 0 \Longrightarrow \lambda\partial|\beta| = \{\lambda\}$ and we have

$$\begin{aligned} r - \beta(\lambda) &= \lambda \\ r - (r - \lambda) &= \lambda \end{aligned}$$

If $r < \lambda$, we set $\beta(\lambda) = r + \lambda$ which again solves (7) because $\beta(\lambda) = r + \lambda < 0 \Longrightarrow \lambda\partial|\beta| = \{-\lambda\}$ and we have

$$\begin{aligned} r - \beta(\lambda) &= -\lambda \\ r - (r + \lambda) &= -\lambda \end{aligned}$$

Case 2: $|r| \leq \lambda$

Since we have $-\lambda \leq r \leq \lambda$, there exist $-1 \leq t_0 \leq 1$ such that $r = t_0\lambda$. We now set $\beta(\lambda) = 0$ and notice that condition (7) becomes

$$r \in \{t\lambda : -1 \leq t \leq 1\}$$

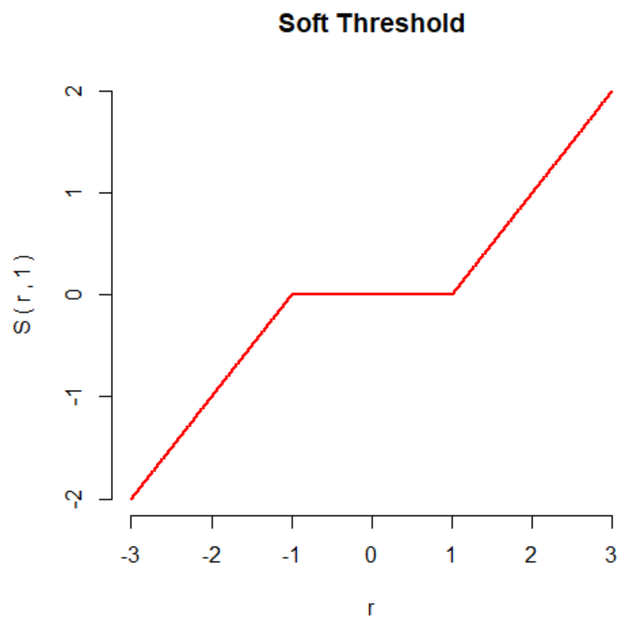
which is true with $t = t_0$. Hence, $\beta(\lambda) = 0$ minimizes $J(\beta, \boldsymbol{\lambda})$.

The Soft Threshold Operator

The solution to condition (7) can be elegantly expressed using the "soft threshold operator" which is given by the function

$$S(r,\lambda) = sign(r)\left(|r| - \lambda\right)^{+} = \begin{cases} r + \lambda & \text{if} \quad r < -\lambda \\ 0 & \text{if} \quad -\lambda \leq r \leq \lambda \\ r - \lambda & \text{if} \quad r > \lambda \end{cases}$$

A plot of $S(r,1)$, as a function of r, with λ equal to one is given below.



Computing Algorithm for the LASSO

Based on the discussion in the previous section, we can implement the following coordinate-descent computing algorithm for the LASSO regression estimate.

1. **Input. Data:** $(y_i, \mathbf{x}_i)$ $\; i = 1, ..., n$ with all the measurements centered and scaled so that

$$\sum y_i = 0, \quad \sum \mathbf{x}_i = \mathbf{0}$$

and

$$\sum y_i^2 = 0, \quad \sum \mathbf{x}_i^2 = \mathbf{1}$$

where, naturally, $(a_1, ..., a_p)^2 = \left(a_1^2, ..., a_p^2\right)$.

Absolute error: $\delta > 0$.

2. **Initialization.** $\boldsymbol{\beta}^0 = (0, ..., 0) \in R^p$

3. **Iteration.** While

$$\left\| \boldsymbol{\beta}^{k+1} - \boldsymbol{\beta}^k \right\| > \delta,$$

given $\boldsymbol{\beta}^k = \left(\beta_1^k, ..., \beta_p^k\right)$ compute $\boldsymbol{\beta}^{k+1} = \left(\beta_1^{k+1}, ..., \beta_p^{k+1}\right)$ as follows:

$$\beta_j^{k+1} = S\left(r_j^k, \lambda\right), \quad j = 1, ..., p$$

with

$$r_j^k = \sum_{i=1}^{n} x_{ij}\widetilde{y}_{ij}$$

and

$$\begin{aligned}
\boldsymbol{\beta}_1^k &= \left(0, \beta_2^k, \beta_3^k, ..., \beta_{p-1}^k, \beta_p^k\right), \quad \widetilde{y}_{i1} = y_i - \mathbf{x}_i'\boldsymbol{\beta}_1^k \\
\boldsymbol{\beta}_2^k &= \left(\beta_1^{k+1}, 0, \beta_3^k, ..., \beta_{p-1}^k, \beta_p^k\right), \quad \widetilde{y}_{i2} = y_i - \mathbf{x}_i'\boldsymbol{\beta}_2^k \\
\boldsymbol{\beta}_3^k &= \left(\beta_1^{k+1}, \beta_2^{k+1}, 0, ..., \beta_{p-1}^k, \beta_p^k\right), \quad \widetilde{y}_{i3} = y_i - \mathbf{x}_i'\boldsymbol{\beta}_3^k \\
\\
\boldsymbol{\beta}_p^k &= \left(\beta_1^{k+1}, \beta_2^{k+1}, \beta_3^{k+1}, ..., \beta_{p-1}^{k+1}, \beta_p^k\right), \quad \widetilde{y}_{ip} = y_i - \mathbf{x}_i'\boldsymbol{\beta}_p^k
\end{aligned}$$

4. **Output**. The pair $\left(\lambda, \boldsymbol{\beta}^{k+1}\right)$, with $\boldsymbol{\beta}^{k+1} = \left(\beta_1^{k+1}, ..., \beta_p^{k+1}\right)$.

Crossvalidation: The choice of λ is made by crossvalidation, using an algorithm similar to that described for ridge regression earlier on.