10.15 Variable Selection and the Bias-Variance Trade-Off

2020-10-02

1 Introduction

We consider the linear model with a univariate responce and p covariates, $Y = X\beta + \epsilon, \epsilon \sim N(0, \sigma^2)$. We denote a training dataset by $\mathcal{T} = (y_t, x_t)_{t=1:N}$ and the least squares estimate of β is the minimiser of the RSS (residual sum of squares) over the training set, $\hat{\beta}^{LS}(\mathcal{T}) = (\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\mathbf{y}$, where x_t is a $1 \times p$ row vector representing and observation. Alternatively, we may employ matrix notation, letting $\mathbf{y} = (y_t)_{t=1:N}$ be a row vector and $\mathbf{x} = (x_{ti})$ an $N \times p$ matrix where each row represents an observation.

Under the assumption $N > p + 1$ we will refer to a subset of the covariates as a *model* $\mathcal{M} \subseteq \{1, ..., p\}$. The LS estimate for $\mathcal{M}$ is computed on a reduced dataset $\mathcal{T}^{\mathcal{M}} = (y_t, (x_{ti})_{i \in \mathcal{M}})_{t=1:N}$. For computational ease we will instead represent the LS estimate for $\mathcal{M}$ in the p-dimensional space of the original model. We denote this representation by $\hat{\beta}^{\mathcal{M}}$, where

$$\hat{\beta}^{\mathcal{M}}_j(\mathcal{T}) = \begin{cases} \hat{\beta}^{\mathcal{M}}_{\pi(j)}(\mathcal{T}^{\mathcal{M}}), & \text{if } j \in \mathcal{M} \\ 0, & \text{otherwise.} \end{cases}$$

Here $\pi : \mathcal{M} \to 1, ..., ||\mathcal{M}||$ maps indicies of covariates in the model to their respectve indicies in the reduced dataset $\mathcal{T}^{\mathcal{M}}$, so that $\mathbf{x}\hat{\beta}^{\mathcal{M}}(\mathcal{T})$ is a notation for $\mathbf{x}^{\mathcal{M}}\hat{\beta}^{LS}(\mathcal{T}^{\mathcal{M}})$.

2 The Bias-Variance tradeoff

2.1 Question 1

We start with model (1) and show the expected squared predictor error at a fixed, arbitrary location u can be deposed as follows ($y \perp \mathcal{T}$):

$$\begin{aligned}\mathbb{E}_{\mathcal{T},y|u}(y-u\hat{\beta})^2 &= \mathbb{E}_{\mathcal{T},y|u}(y^2 - 2yu\hat{\beta} + (u\hat{\beta})^2) \\ &= \mathrm{Var}_{\mathcal{T},y|u}(y) + (u\beta)^2 - 2u\beta\mathbb{E}_{\mathcal{T}}(u\hat{\beta}) + (\mathbb{E}_{\mathcal{T}}(u\hat{\beta}))^2 + \mathrm{Var}_{\mathcal{T}}(u\hat{\beta}) \\ &= \sigma^2 + (u\beta - \mathbb{E}_{\mathcal{T}}(u\hat{\beta}))^2 + \mathrm{Var}_{\mathcal{T}}(u\hat{\beta}).\end{aligned}$$

The summands on the right-hand side are reffered to as the *irreducible variance*, *squared estimation bias* (SEB) and *estimation variance*, respectively. Now consider the effect of removing the p^{th} covariate on each of these for $\hat{\beta} = \hat{\beta}^{LS}$. Under the assumption that $\mathbf{x}^T\mathbf{x} = I_p$ we see $\mathbf{x}_{\mathcal{M}}^T\mathbf{x}_{\mathcal{M}} = I_{p-1}$, so $u\hat{\beta} = u\hat{\beta}^{\mathcal{M}} + u_p\hat{\beta}^{\mathcal{M}}$. Clearly, σ remains unchanged. Now for the SEB:

$$\begin{aligned}SEB_{\mathcal{T}^{\mathcal{M}}} &= (u\beta - \mathbb{E}_{\mathcal{T}^{\mathcal{M}}}(u\hat{\beta}^{\mathcal{M}}))^2 \\ &= (u\beta - \mathbb{E}_{\mathcal{T}}(u\hat{\beta}) + \mathbb{E}_{\mathcal{T}}(u_p\hat{\beta}_p))^2 \\ &= \mathbb{E}_{\mathcal{T}}(u_p\hat{\beta}_p)^2 \\ &\geq 0 = SEB_{\mathcal{T}} \text{ (by the fact bias}(\hat{\beta}^{LS}) = 0),\end{aligned}$$

with equality if and only if $\beta_p = 0$. For the estimation variance (note if a, b independant, then $\mathbb{E}(ab) = \mathbb{E}(a)\mathbb{E}(b)$):

$$\begin{aligned}\mathrm{Var}_{\mathcal{T}^{\mathcal{M}}}(u\hat{\beta}^{\mathcal{M}}) &= \mathrm{Var}_{\mathcal{T}^{\mathcal{M}}}(u\hat{\beta} - u_p\hat{\beta}_p) \\ &= \mathbb{E}\left[\left(u\hat{\beta} - u_p\hat{\beta}_p - \mathbb{E}(u\hat{\beta} - u_p\hat{\beta}_p)\right)^2\right] \\ &= \mathrm{Var}_{\mathcal{T}}(u\hat{\beta}) - 2\mathbb{E}\left[(u\hat{\beta} - \mathbb{E}(u\hat{\beta}))(u_p\hat{\beta}_p - \mathbb{E}u_p\hat{\beta}_p)\right] + \mathrm{Var}_{\mathcal{T}}(u_p\hat{\beta}_p) \\ &= \mathrm{Var}_{\mathcal{T}}(u\hat{\beta}) - 2\mathbb{E}\left[\left(\sum_{i=1}^{p} u_i\hat{\beta}_i\right)\left(u_p\hat{\beta}_p - \mathbb{E}u_p\hat{\beta}_p\right)\right] + \mathrm{Var}_{\mathcal{T}}(u_p\hat{\beta}_p)\end{aligned}$$

As we assume $\mathbf{x}^T\mathbf{x} = I_p$ then, $\hat{\beta}_i, \hat{\beta}_j$ indep for $i \neq j$.

$$\begin{aligned}\mathrm{Var}_{\mathcal{T}^{\mathcal{M}}}(u\hat{\beta}^{\mathcal{M}}) &= \mathrm{Var}_{\mathcal{T}}(u\hat{\beta}) - 2\mathbb{E}\left[(u_p\hat{\beta}_p)(u_p\hat{\beta}_p - \mathbb{E}u_p\hat{\beta}_p)\right] + \mathrm{Var}_{\mathcal{T}}(u_p\hat{\beta}_p) \\ &= \mathrm{Var}_{\mathcal{T}}(u\hat{\beta}) - \mathrm{Var}_{\mathcal{T}}(u_p\hat{\beta}_p)\end{aligned}$$

So, $\mathrm{Var}_{\mathcal{T}^{\mathcal{M}}}(u\hat{\beta}^{\mathcal{M}}) \leq \mathrm{Var}_{\mathcal{T}}(u\hat{\beta})$, with equality if and only if $\mathrm{Var}_{\mathcal{T}}(u_p\hat{\beta}_p) = 0$. Thus, removing the p^{th} covariate will increase the bias but decrease the variance. Combining the results above we see $\mathbb{E}_{\mathcal{T}^{\mathcal{M}},y|u}(y - u\hat{\beta}^{\mathcal{M}})^2 = \mathbb{E}_{\mathcal{T},y|u}(y - u\hat{\beta})^2 + \left(\mathbb{E}(u_p\hat{\beta}_p)\right)^2 \leq \mathbb{E}_{\mathcal{T},y|u}(y - u\hat{\beta})^2$, and so the expected square error also falls, when removing the p^{th} co-variate.

2.2 Question 2

Use the same model as above with $p = 10, \sigma^2 = 1$, and $X \sim N(0, I_p)$, and set $\beta = (-0.5, 0.45, -0.4, 0.35, -0.3, 0.25, -0.2, 0.15, -0.1, 0.05)^T$. We then consider $\mathcal{M}_1 = \{1\}, ..., \mathcal{M}_p = \{1, ..., p\}$, trained on a data set size N_{tr} and test data set of size $N_{te} = 1000$. Program 1 (reference completes this simulation and produces plots of the relative training and testing RSS averaged over 100 experiments graphed against the respective M_j. The following shows this run for $N_{tr} = 30$ and 200 with the respective tables below and graphs produced in Figures 1 and 2.

Listing 1: Completing statistical training 1

```
>>>true_beta = [-0.5,0.45,-0.4,0.35,-0.3,0.25,-0.2,0.15,-0.1,0.05]
>>>train_and_test(30, 1000, 10, 1, true_beta)
```

M_j	Average Testing RSS	Average Training RSS
1	1.7763	1.6314
2	1.6015	1.4006
3	1.4854	1.2210
4	1.4021	1.0673
5	1.3570	0.9589
6	1.3629	0.8529
7	1.3845	0.7851
8	1.4129	0.7283
9	1.4658	0.6819
10	1.5120	0.6482

Listing 2: Completing statistical training 2

```
>>>train_and_test(200, 1000, 10, 1, true_beta)
```

M_j	Average Testing RSS	Average Training RSS
1	1.7269	1.6973
2	1.5130	1.4946
3	1.3620	1.3325
4	1.2584	1.2071
5	1.1616	1.1021
6	1.1046	1.0356
7	1.0726	0.9910
8	1.0620	0.9667
9	1.0547	0.9531
10	1.0456	0.9474

It is first important to note that $RSS(\hat{\beta};\mathcal{T}) = \frac{1}{N}\sum_{t=1}^{N}(y_t - x_t\hat{\beta})^2$, so in the ideal scenario as $\hat{\beta} \to \beta$, $\mathbb{E}(RSS) \to \mathbb{E}(\epsilon^2) = \sigma^2 = 1$. The average RSS values approximate $\mathbb{E}(RSS)$, so it is clear that in both cases we were seeing over training of the data for high j as these give a training RSS less than 1. The negative affects of this on the accuracy of the model on test data can be seen in the respective tables and plots which show the optimal testing RSS values to occur for $j = 5$ and $j = 10$ for $N_{tr} = 30$ and 200 respectively. The first shows consistency with the results of Question 1 (this time for a less restricted model, $X^TX \neq I_p$ necessarily) where a balance of bias and variance must be met by managing the number of co-variate terms considered. It is also worth noting that in the $N_{tr} = 200$ case we see little benefit by considering fewer co-variates and that for $j > 6$ the models give very close approximations of the optimal expected RSS. This is because with this larger training data size our program is able to better estimate β, ignoring the noise produced by ϵ thus, avoiding over training. We then see as expected this increases the training RSS and reduces the testing RSS, with both approaching 1.

3 Variable selection methods

3.1 Subset Selection

3.1.1 Question 3: Best subsets selection

Program 2 shows a procedure which takes the input of a training dataset $\mathcal{T}$ and outputs a $p \times p$ matrix B, whose j^{th} column contains $\hat{\beta}^{\mathcal{M}_j}$ for the best (RSS) performing model of size j. The program assumes the same model as used within Question 2 but with some slight tweaks could be made to deal with any more general linear model. To run an example below the 30×10 training matrix was first formed as before then used as the input for Program 2 to produce the following:

Listing 3: Best subsets selection on Question 2 model and training dataset

```
>>>training_dataset = form_list(30,10)
>>>bestsubset(training_dataset)
#Output matrix B transcribed into nicer format below
```

$$B = \begin{bmatrix} -0.44 & -0.40 & -0.48 & -0.52 & -0.44 & -0.57 & -0.52 & -0.47 & -0.47 & -0.41 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -0.02 & -0.01 \\ 0 & 0 & -0.64 & -0.58 & -0.67 & -0.58 & -0.55 & -0.60 & -0.60 & -0.63 \\ 0 & 0 & 0 & 0.15 & 0.11 & 0.25 & 0.31 & 0.27 & 0.27 & 0.24 \\ 0 & 0 & 0 & -0.33 & -0.29 & -0.30 & -0.27 & -0.24 & -0.25 & -0.20 \\ 0 & 0.16 & 0 & 0 & 0.28 & 0 & 0 & 0.16 & 0.15 & 0.17 \\ 0 & 0 & 0 & 0 & 0 & -0.11 & -0.11 & -0.03 & -0.03 & -0.03 \\ 0 & 0 & 0 & 0 & 0 & 0.30 & 0.36 & 0.30 & 0.30 & 0.33 \\ 0 & 0 & 0 & 0 & 0 & 0 & -0.19 & -0.19 & -0.19 & -0.24 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -0.15 \end{bmatrix}$$

$$RSS = \begin{bmatrix} 1.66 & 1.64 & 1.48 & 1.28 & 1.34 & 1.34 & 1.30 & 1.29 & 1.26 & 1.37 \end{bmatrix}$$

The RSS vector was also produced to give insight into the relative optimal accuracy for each value of j, and showed consistency with the theory discussed in Question 1 and results of Question 2. Interestingly, B is nearly diagonal which supports what we would expect seeing as the terms of β decrease in magnitude for increasing indexes (discrepancies arise from the stochastic nature of the program). It is important to note that this method is very computationally expensive, with a model space of size $|\{\mathcal{M} : \mathcal{M} \subseteq \{1, ..., p\}| = 2^p - 1$ (formed on the logic that creating a subset has two options for each element of the set, to include or not include in the subset, then ignore the empty set) individual cases to find the optimal $\mathcal{M}_j$ for all j. Set $K_j = |\{\mathcal{M} : \mathcal{M} \subseteq \{1, ..., p\}, ||\mathcal{M}|| = j\}|$ for $j \in \{1, ..., p\}$, we know the program runs under acceptable time frames for $\max_j(K_j) < 10^5$. Then note by the combinatoric definition of the binomial coefficient we see $K_j = \binom{p}{j}$. It is clear that this is at a maximum for $j = \lfloor \frac{p}{2} \rfloor$ by considering the equality $\binom{n}{r+1} = \frac{n-r}{r+1}\binom{n}{r}$, and the respective values of r for which $\binom{n}{r}$ increases/decreases. Then note, $\binom{19}{9} = 92378 < 10^5 < 184756 = \binom{20}{10}$, thus the smallest such p is 20.

3.1.2 Question 4: Greedy subset selection

We now look at a procedure *greedysubset*, using the same input-output format as before, that incrementally builds the model sequence $\mathcal{M}_j$ at each iteration as follows:

$$\mathcal{M}_0 = \varnothing, \mathcal{M}_{d+1}(\mathcal{T}) = \mathcal{M}_d(\mathcal{T}) \cup \left\{ l : l = \arg\min_j RSS\left(\hat{\beta}^{\mathcal{M}_d(\mathcal{T}) \cup \{j\}}(\mathcal{T}); \mathcal{T}\right) \right\}.$$

This procedure is implemented in Program 3, producing the following:

Listing 4: Greedy subsets selection on Question 2 model and training dataset

```
>>>training_dataset = form_list(30,10)
>>>greedysubset(training_dataset)
#Output matrix B transcribed into nicer format below
```

$$B = \begin{bmatrix} -0.38 & -0.37 & -0.39 & -0.46 & -0.41 & -0.43 & -0.34 & -0.41 & -0.43 & -0.43 \\ 0 & 0.46 & 0.49 & 0.45 & 0.41 & 0.39 & 0.38 & 0.39 & 0.46 & 0.45 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.14 & 0.12 & 0.11 \\ 0 & 0 & 0.12 & 0.17 & 0.21 & 0.12 & 0.12 & 0.38 & 0.29 & 0.26 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.30 \\ 0 & 0 & 0 & 0.27 & 0.30 & 0.29 & 0.27 & 0.32 & 0.30 & -0.20 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -0.21 & 0.30 \\ 0 & 0 & 0 & 0 & 0.35 & 0.31 & 0.31 & -0.24 & 0.30 & 0.14 \\ 0 & 0 & 0 & 0 & 0 & -0.28 & -0.18 & 0.28 & 0.15 & -0.17 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.30 & 0.22 & -0.17 & 0.04 \end{bmatrix}$$

$$RSS = \begin{bmatrix} 1.64 & 1.44 & 1.43 & 1.32 & 1.34 & 1.35 & 1.52 & 1.81 & 1.59 & 2.08 \end{bmatrix}$$

Again the RSS vector is also provided as it gives a good insight into the accuracy of this model. The first thing to note is the algorithm is much faster in producing the matrix B. However, we this comes at a cost as we see the solution stray further from the upper triangular matrix we know is the optimal solution and the respective RSS values on average appear slightly higher than those found by *bestsubset*.

We note that the computational complexity could be even further improved upon by noting the the family of models is nested allowing us to more efficiently calculate $(X^T_{\mathcal{M}_{j+1}} X_{\mathcal{M}_{j+1}})^{-1}$, normally being $\mathcal{O}(N^2_{tr}(j+1)^2)$ for the multiplication and then $\mathcal{O}((j+1)^3)$ for the inverse. Then work from the assumption $M_j = \{1, ..., j\}$ (if not, the following still applies simply under a change of basis):

$$X = X^T_{\mathcal{M}_{j+1}} X_{\mathcal{M}_{j+1}} = \begin{bmatrix} X^T_{\mathcal{M}_j} X_{\mathcal{M}_j} & X^T_{\mathcal{M}_j} x_{j+1} \\ x^T_{j+1} X_{\mathcal{M}_j} & |x_{j+1}|^2 \end{bmatrix},$$

$$X' = \begin{bmatrix} A(X^T_{\mathcal{M}_j} X_{\mathcal{M}_j})^{-1} & \mathbf{b} \\ \mathbf{a}^T & c \end{bmatrix},$$

with A some $j \times j$ matrix, $\mathbf{a}, \mathbf{b} \in \mathbb{R}^j$ and $c \in \mathbb{R}$. Set $X'X = I_{j+1}$, giving:

$$A + \mathbf{b} x^T_{j+1} X_{\mathcal{M}_j} = I_j \tag{1}$$

$$A(X^T_{\mathcal{M}_j} X_{\mathcal{M}_j})^{-1} X^T_{\mathcal{M}_j} x_{j+1} + \mathbf{b}|x_{j+1}|^2 = \mathbf{0} \tag{2}$$

$$\mathbf{a}^T X^T_{\mathcal{M}_j} X_{\mathcal{M}_j} + c x^T_{j+1} X_{\mathcal{M}_j} = \mathbf{0}^T \tag{3}$$

$$\mathbf{a}^T X^T_{\mathcal{M}_j} x_{j+1} + c|x_{j+1}|^2 = 1 \tag{4}$$

Now this can be solved for $A(X^T_{\mathcal{M}_j} X_{\mathcal{M}_j})^{-1}, \mathbf{a}, \mathbf{b}$ and c, assuming the existence of a solution (which occurs almost surely), starting with (1) and (3) for A and $\mathbf{a}$ respectively, then using (2) and (4) to give $\mathbf{b}$ and c in terms of $X_{\mathcal{M}_j}, x_{j+1}$ and $(X^T_{\mathcal{M}_j} X_{\mathcal{M}_j})^{-1}$.

$$c = \frac{1}{|x_{j+1}|^2 - x^T_{j+1} X_{\mathcal{M}_j} (X^T_{\mathcal{M}_j} X_{\mathcal{M}_j})^{-1} X^T_{\mathcal{M}_j} x_{j+1}}$$

$$\mathbf{b} = -c(X_{\mathcal{M}_j}^T X_{\mathcal{M}_j})^{-1} X_{\mathcal{M}_j}^T x_{j+1}$$
$$A(X_{\mathcal{M}_j}^T X_{\mathcal{M}_j})^{-1} = (X_{\mathcal{M}_j}^T X_{\mathcal{M}_j})^{-1} - \mathbf{b}x_{j+1}^T X_{\mathcal{M}_j}(X_{\mathcal{M}_j}^T X_{\mathcal{M}_j})^{-1}.$$
$$\mathbf{a} = -cx_{j+1}^T X_{\mathcal{M}_j}(X_{\mathcal{M}_j}^T X_{\mathcal{M}_j})^{-1}$$

This reduces the complexity to $\mathcal{O}(N_{tr}j + j^3)$, which is particularly significant for large N_{tr}.

3.1.3 Question 5: Forward F-test

We now want to consider the use of an F-statistic ($F \sim F_{1,pN_{tr}-d-1}$)

$$F = \frac{RSS(\hat{\beta}^{\mathcal{M}_d}) - RSS(\hat{\beta}^{\mathcal{M}_{d+1}})}{RSS(\hat{\beta}^{\mathcal{M}_{d+1}})/(pN_{tr} - d - 1)}$$

to force our program to stop when increasing d does not significantly improve the fit at the $p = 0.05$ level. This is implemented in *greedysubset2* (Program 4), with use of the *scipy.stats.f.cdf* function in python. Program 4 is set up return matrix B however, for clarity below only the final vector of $\hat{\beta}^{\mathcal{M}_j}$ and its RSS will be displayed. It is also important to note, in the case that $RSS(\hat{\beta}^{\mathcal{M}_j}) < RSS(\hat{\beta}^{\mathcal{M}_{j+1}})$ the algorithm will terminate and return B.

Listing 5: Greedy subsets 2 selection on Question 2 model and training dataset

```
>>>training_dataset = form_list(30,10)
>>>greedysubset2(training_dataset)
RSS = 1.332754536504679
[-0.61, 0.54, 0, 0.37, 0, 0, 0, 0, 0, 0]
```

This same method could not be used in for the best subset selection method, this is because in the general case F no longer follows an $F \sim F_{1,pN_{tr}-d-1}$ distribution. We can see this by considering $RSS(\hat{\beta}^{\mathcal{M}_d}) = ||(I - P_d)Y||^2$ and $RSS(\hat{\beta}^{\mathcal{M}_{d+1}}) = ||(I - P_{d+1})Y||^2$ for the respective projection matrices P_d and P_{d+1}. The model requires $RSS(\hat{\beta}^{\mathcal{M}_d}) - RSS(\hat{\beta}^{\mathcal{M}_{d+1}}) = ||(P_{d+1} - P_d)Y||^2 \sim \chi_1^2$. This is true only if $P_{d+1} - P_d$ is a projection matrix, but we note that $(P_{d+1} - P_d)^2 = P_{d+1} + P_d$ if $\mathcal{M}_d \cap \mathcal{M}_{d+1} = \varnothing$. So $P_{d+1} - P_d$ is not a projection matrix (as it is not idempotent) and the F test is invalid here.

3.1.4 Question 6

We consider now a new estimate of $\beta \approx \hat{\beta}^{CV}$ which selects one of the solutions of a given sparse estimator ($\hat{\beta}^{(j)}$ is the j^{th} candidate with precisely $p - j$ zeros), chosen on the basis of estimated predicted error $\hat{PE}$:

$$\hat{\beta}^{CV}(\mathcal{T}) = \hat{\beta}^{j^*}(\mathcal{T}), \text{ where } j^* = \arg\min_j \left\{ \hat{PE}(j, \mathcal{T}) \right\}.$$

The prediction error can be estimated using 10-fold cross-validation as

$$\hat{PE}(j,\mathcal{T}) = \frac{1}{10}\sum_{k=1}^{10} RSS\left(\hat{\beta}^{(j)}(\mathcal{T}^{-k});\mathcal{T}^{k}\right),$$

where $\mathcal{T}^k$ is the k^{th} fold of the training set and $\mathcal{T}^{-k}$ its complement:

$$\mathcal{T}^k = \left\{(y_{\pi(n)}, x_{\pi(n)})|k-1 < \frac{10n}{N_{tr}} \leq k\right\}$$
$$\mathcal{T}^{-k} = \left\{(y_{\pi(n)}, x_{\pi(n)})|\frac{10n}{N_{tr}} \leq k-1 \text{ or } \frac{10n}{N_{tr}} > k\right\},$$

where $\pi(n)$ is a random permutation of $\{1, ..., N_{tr}\}$, calculated uniquely for each k, each run through the algorithm (using *np.random.permutation*). The running of this algorithm requires this procedure to take a function as an input argument, in python this is no different to using a variable as it treats both objects very similarly. This is put together in Program 5 (REFERENCE), taking an input $\mathcal{T}$ and a sparse estimator, and giving the output $\hat{\beta}^{CV}(\mathcal{T})$. Some examples read below:

Listing 6: Finding the cross-validation estimator for beta

```
>>>training_dataset = form_list(30,10)
>>>crossval(training_dataset, greedysubset)
array([-0.36,  0.51,  0,  0,  0,  0,  0,  0,  0,  0])
>>>crossval(training_dataset, bestsubset) #long run time
array([-0.41,  0.48, -0.36,  0.13, -0.26,  0, -0.21,  0,  0.08,  0])
```

3.2 The Lasso estimator

3.2.1 Question 7

The lasso estimator penalises the RSS as follows:

$$\hat{\beta}^{(L,\lambda)}(\mathcal{T}) = \arg\min_{\hat{\beta}}\left\{RSS\left(\hat{\beta};\mathcal{T}\right) + \lambda\sum_{j=1}^{p}\left|\hat{\beta}_j\right|\right\}. \tag{5}$$

This can be transformed into a quadratic program with linear constraints as follows (using the fact that λ is fixed),

$$\hat{\beta}^{(L,\lambda)}(\mathcal{T}) = \arg\min_{\hat{\beta}}\left\{RSS\left(\hat{\beta};\mathcal{T}\right) + \lambda\left(\sum_{j=1}^{p}\left|\hat{\beta}_j\right| + z(\lambda) - k(\lambda)\right) : z,k \geq 0\right\}$$

We recognise this as the Lagrangian for the problem ($\mathcal{T} = (\mathbf{y}, \mathbf{x})$):

$$\begin{aligned}\hat{\beta}^{(L,\lambda)}(\mathcal{T}) &= \arg\min_{\hat{\beta}} \left\{ (\mathbf{y} - \mathbf{x}\hat{\beta})^T(\mathbf{y} - \mathbf{x}\hat{\beta}) : \sum_{j=1}^{p} \left|\hat{\beta}_j\right| \leq k(\lambda) \right\} \\ &= \arg\min_{\hat{\beta}} \left\{ \hat{\beta}^T(\mathbf{x}^T\mathbf{x})\hat{\beta} - (2\mathbf{y}^T\mathbf{x})\hat{\beta} : \sum_{j=1}^{p} \left|\hat{\beta}_j\right| \leq k(\lambda) \right\}\end{aligned}$$

Now to linearise the constraints we must remove the absolute values, this can be done using the $2^p \times p$ matrix A with rows consisting of all permutations of ± 1's:

$$A = \begin{bmatrix} 1 & \dots & 1 & -1 \\ 1 & \dots & -1 & 1 \\ 1 & \dots & -1 & -1 \\ \vdots & & \vdots & \vdots \\ -1 & \dots & -1 & -1 \end{bmatrix}$$

Noting $|a| = \max\{a, -a\}$, we arrive at the required quadratic program with linear constraints as follows:

$$\hat{\beta}^{(L,\lambda)}(\mathcal{T}) = \arg\min_{\hat{\beta}} \left\{ \hat{\beta}^T(\mathbf{x}^T\mathbf{x})\hat{\beta} - (2\mathbf{y}^T\mathbf{x})\hat{\beta} : A\hat{\beta} \leq k(\lambda) \right\}.$$

3.3 Question 8

4 Appendices

4.1 Figures

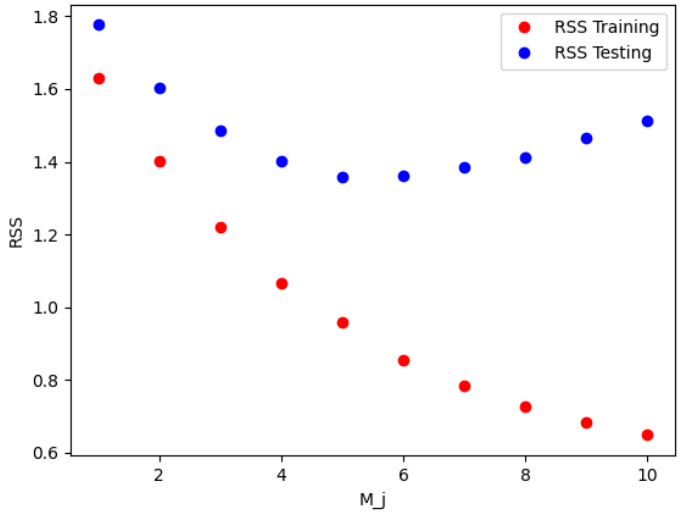

Figure 1: Average RSS in training and test data sets for LS estimator with $N_{tr} = 30$ and $N_{te} = 1000$.

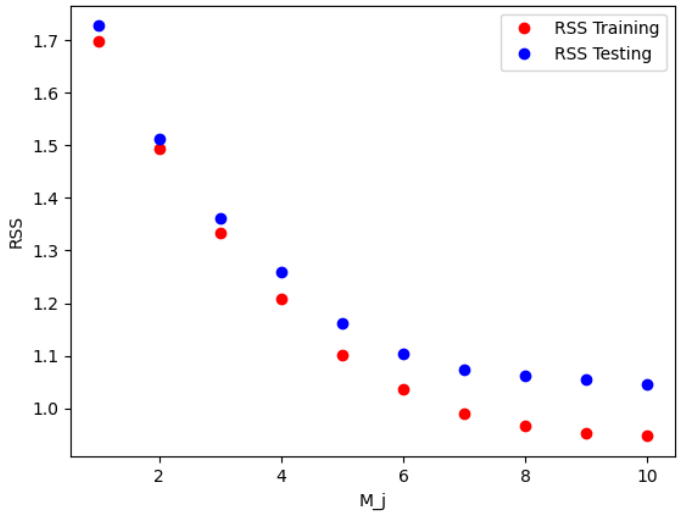

Figure 2: Average RSS in training and test data sets for LS estimator with $N_{tr} = 200$ and $N_{te} = 1000$.