ST443 Group Project Part 2

December 10, 2020

We aim to look at the variable selection and regularisation methods LASSO (least absolute shrinkage and selection operator) and Elastic Net (EN) and compare their results. The methods differ in the penalty term they include. LASSO uses the l_1 norm and EN uses the l_2 norm in addition to the l_1 norm.

We aim to show that the coordinate descent algorithm can be used to find the optimal values of the penalty terms in the LASSO and Elastic Net algorithms, and the accompanying β's. We denote the penalty terms as λ where $\hat{\lambda}_1$ is the penalty term for LASSO and $\hat{\lambda}_2$ is the additional penalty term in Elastic Net, as well as $\hat{\lambda}_1$. These penalty terms will shrink the coefficients (β's) down towards zero, and many of them can become exactly 0. Because the Elastic Net combines two penalty terms, it tends to shrink the coefficients less than LASSO and thus has less coefficients equalling 0. Also, since the Elastic Net uses the l_2 norm the coefficients will always be positive (as the l_2 norm includes a squared term).

Dataset Creation

We created the dataset by taking random samples from $N \sim (0, I_N)$ for each of our X values and repeated the same process once for a value of ϵ which was our noise vector. These were inputs to our model $\boldsymbol{Y} = \boldsymbol{\beta} X + \sigma \boldsymbol{\epsilon}$.

Initially we used the parameter values taken from the question brief as below:

- $\sigma = 3$
- Actual β's = (3, 1.5, 0, 0, 2, 0, 0, 0)
- Number of datasets = 50
- Number of observations, n, in total = 240
- Number of observations in training set = 20
- Number of observations in validation set = 20
- Number of observations in testing set = 200
- Pairwise correlation between each of the variables was calculated as: $corr(i, j) = (0.5)^{i-j}$, denoted cor.

We used nested *for* loops to calculate the pairwise correlations which were saved to a matrix *cor*.

We created the *data_creation* function with arguments $(cor, x, n, \sigma, actual\_beta)$ where *cor*, σ, and *actual_beta* are described as above, n is the total number of observations and x is the proportion of n which is dedicated to the training set, conversely $(1 - x)$ is used inside the function to calculate the proportion dedicated to the validation set. *data_creation* creates the respective independent training, independent validation, and test sets as well as the accompanying noise vectors. Further, we create vectors of the true Y values that would be expected using the actual beta values which will be used to calculate the partial residuals later when estimating the respective β values.

Coordinate Descent Algorithm

Partial Residuals

To estimate the betas we must calculate the partial residuals as per 2(a) in the brief. For each i, j we calculate the residuals using only the values for j that do not match the current j^{th} element we are evaluating.

Least Squares

Using the matrix of partial residuals we estimated the values of each of the betas by taking the sum of the products of every pairwise residual with its corresponding x_{ij} observation which was taken from the training set.

Soft Thresholding

We used soft thresholding to decide if the current estimate of each β is optimal after introducing the relevant penalty terms, λ_1 only for LASSO and both λ_1 and λ_2 for the Elastic Net. The soft thresholding is implemented as:

$$\beta_j = sign(\beta_j^*)max((|\beta_j^*| - \lambda_1), 0)(1 + 2\lambda_2)^{-1}$$

This formula can be used equally in the LASSO Regression and the Elastic Net, since the only difference between the two is the addition of the penalisation with λ_2 in the Elastic Net. In case we are doing LASSO, λ_2 would be equal to 0, and the formula would do the soft thresholding update for LASSO.

Convergence

We repeat this cycle until convergence. Inside our *coord_descent* function we included an $ifelse$ condition to check if there has been a significant change in our β's. At the point where there is no change in the first 5 decimal places of each β we class this as convergence and record the resulting β's.

Model Validation

In order to select the optimal λ_1, and, if applicable, λ_2, for our model, we ran a validation with our self-generated data. For each model, we determined the β coefficients through the coordinate descent algorithm for a range of λ from 0 to 4. We then computed the validation MSE, and selected the λ penalization terms that returned the lowest MSE as the optimal values.

We first did this by creating a sequence with R's built-in *seq*() function from 0 to 4, storing all the validation MSEs with their respective λ and selecting the minimum MSE. Then we compared our results with the optimization function *optim*() on the validation MSE. Because of computational limitations we can choose relatively few values for λ, all of which are evenly spaced as per R's $runif$() function. The *optim*() function returned the global minimum, and the difference to our result was insignificant. Due to the computational cost of using the *optim*() function, we decided to proceed with our self-developed function for the rest of the project.

RandomSearch is computationally more efficient than GridSearch, and also more statistically robust. Hence we produced our final λ range by sampling from a uniform distribution from 0 to 4, which gave more precise values, similar to the optimization function and as opposed to the initial *seq*() function. Figure 1 shows the relationship between the $\hat{\lambda}_1$ penalty

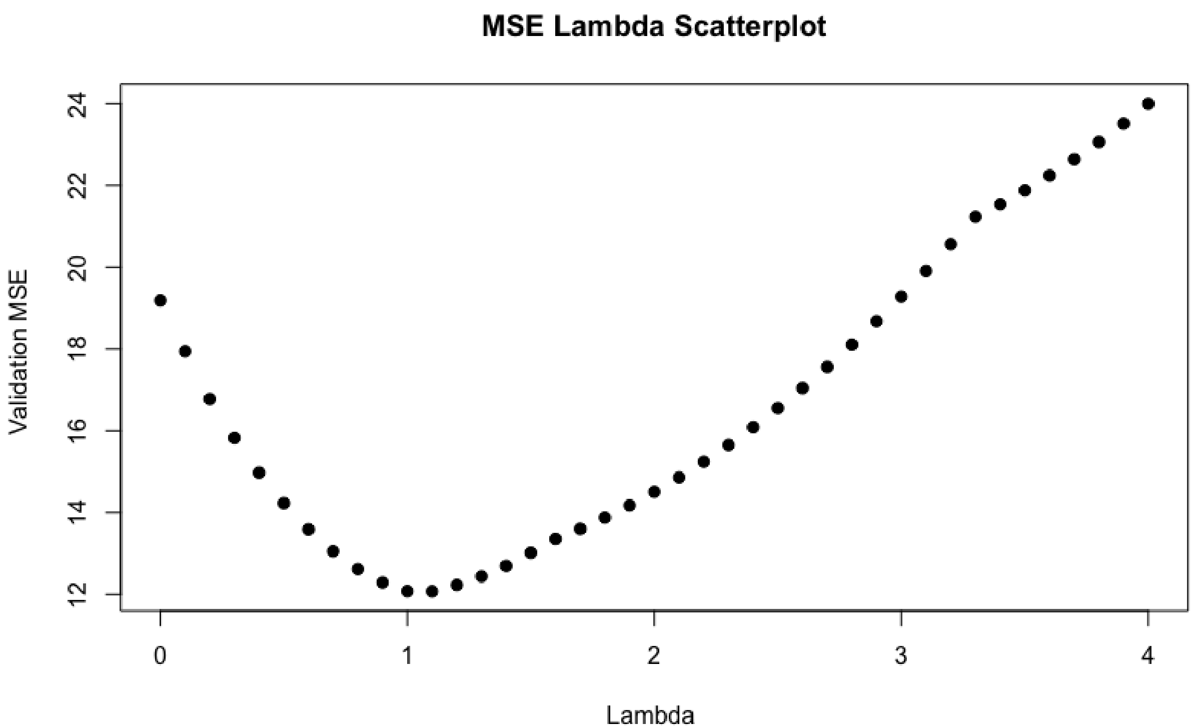

Figure 1: Validation MSE vs. corresponding λ_1 for the LASSO Regression

term for LASSO and the corresponding validation MSE. For the model given in the instructions, a value of approximately 1 resulted in the lowest validation MSE of approximately

12.

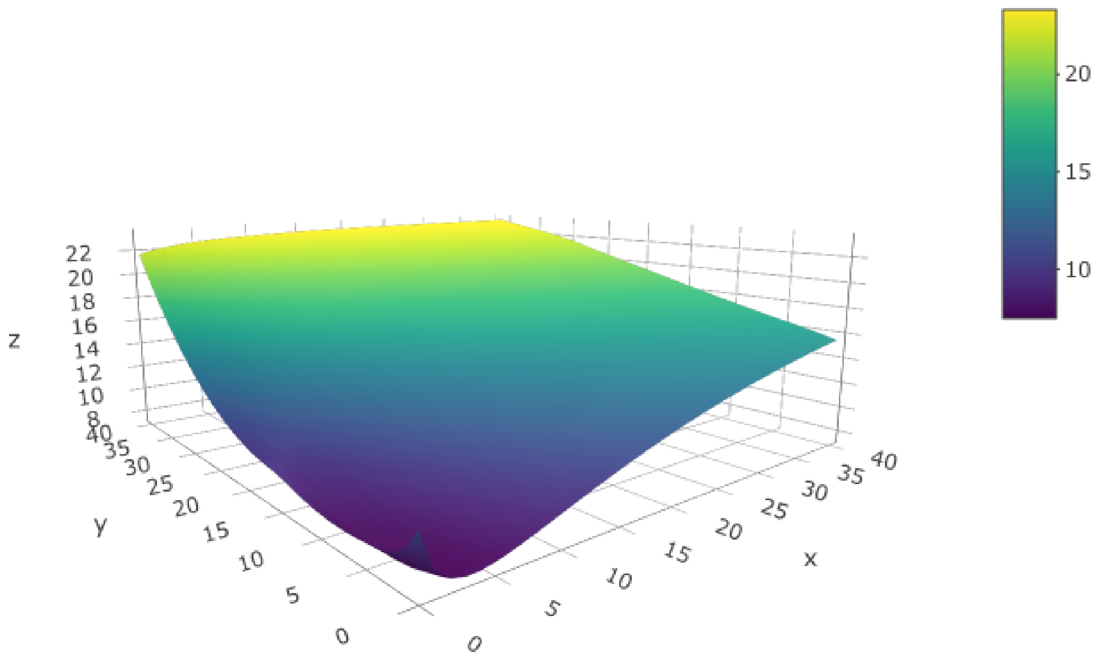

Figure 2: Validation MSE vs. corresponding λ_1 and λ_2 for the Elastic Net Regression

Figure 2 shows the relationship between the $\hat{\lambda}_1$ and $\hat{\lambda}_2$ penalty terms for the Elastic Net Regression, and the corresponding validation MSE. For the model given in the instructions, values of approximately 0.1 for $\hat{\lambda}_1$ and 0.3 for $\hat{\lambda}_2$ resulted in the lowest validation MSE of approximately 7.5.

After performing the validation we can already see that the Elastic Net has a better prediction capacity, as its minimum MSE is below substantially lower than the one of LASSO.

Simulations

Initially we used the default values described in the instructions and specified again above. Table shows the results for both LASSO and Elastic Net.

Testing Results	Lasso			Elastic Net		
n	MSE	MSE st. error	# vars	MSE	MSE st error	# vars
200	14.57	4.12	3.99	13.32	3.04	5.76

Consistently with what we saw in the validation errors, the Elastic Net has a better performance than LASSO as measured by the Test MSE, although the difference in this case seems to be lower than in the validation. Moreover, the Elastic Net appears to be more consistent as its Test MSE has a lower standard deviation (shown by the MSE st. error). We can also see that LASSO shrinks the variables more, since on average it produces models with 4 variables, whereas the Elastic Net produces ones with close to 6. In the following we will examine the model results after changing the initial parameters to compare both in more detail.

Altering Parameter Values

Changing σ

σ was used to change the level of noise in the Y values. σ is a scalar value of a Normally Distributed noise vector with mean 0 and variance 1. Increasing sigma caused an increase in the noise associated with Y and hence increased variation in our Y values. This in turn caused the average test MSE to increase as σ increased.

LASSO and EN have similar MSE - although those for EN are slightly lower, especially the lower σ. As we saw earlier, EN is considerably better than LASSO when we look at the standard errors on the MSE results, as it has a lower st. error for all levels of sigma. Further to this the # of non-zero vars estimated have a lower st. error for EN than LASSO.

In conclusion, EN performs better than LASSO when there is large variance in the original dataset, which is caused here by scaling the noise vector. The MSE are comparable but the st. error is much lower, which should give us more confidence in EN as a model than LASSO.

	Lasso				Elastic Net			
σ	MSE	MSE st. error	# vars	# vars st. error	MSE	MSE st. error	# vars	# vars st. error
0	2.83	3.02	4.52	1.97	2.27	1.78	6.64	1.42
1	4.08	3.40	4.12	1.80	3.10	1.85	7.06	1.10
2	8.64	4.66	4.80	2.05	7.03	2.09	6.86	1.25
3	14.87	4.00	4.42	1.93	13.03	2.48	7.00	1.47
4	23.60	5.36	3.78	1.90	21.10	3.63	6.26	1.79
5	32.30	6.17	3.82	2.18	31.54	5.72	6.73	1.66
6	44.89	7.36	3.83	2.02	44.22	5.40	6.12	2.16

Table 1: Values were calculated using the *sigma_loop_L* and *sigma_loop_EN* functions.

Changing total n

We can see that as $n \to \infty$ that the difference between LASSO and EN stays consistent, and both their errors decrease. However, we would struggle to run $n > 4000$ as the computation time for $n = 4000$ was approximately 18 minutes. It is likely that test MSE would have continued to decrease for both LASSO and EN, however, we experience diminishing returns. It was often the case that the MSE of LASSO was marginally lower than EN, and in 5 out of 6 cases had a marginally lower st. error also. However, when we look at the number of non-zero vars EN is much more consistent with it's estimations of the number of variables.

	Lasso				Elastic Net			
No of obs (n)	MSE	MSE st. error	# vars	# vars st. error	MSE	MSE st. error	# vars	# vars st. error
40	11.76	2.33	5.34	2.02	11.61	1.70	6.82	1.29
200	9.46	0.43	5.4	1.48	9.82	0.52	7.18	0.69
500	9.18	0.21	5.92	1.88	9.67	0.31	7.14	0.73
1000	9.09	0.15	5.86	2.13	9.27	0.26	7.06	0.79
2000	9.05	0.09	5.48	2.10	9.21	0.17	6.88	0.69
4000	9.02	0.07	6.36	2.29	9.18	0.13	6.88	0.77

Table 2: Values were calculated using the *n_loop_L* and *n_loop_EN* functions.

This may be expected though as EN does not have the ability to reduce variables to exactly zero, like LASSO does.

Changing train/validation proportion split

Initially the number of observations dedicated to training and validation was 20 and 20 respectively out of 240 observations (the remainder were dedicated to the testing set). This gave us a 50/50 split of potentially seen data to use for training and validation. This is not a drastically poor split but the common consensus is to use a 75/25 or 80/20 split instead.

We kept the testing set size consistent through all tests at $10n$ as per the default ratio.

	Lasso			Elastic Net		
train/val split %	MSE	MSE st. error	# vars	MSE	MSE st error	# vars
25/75	18.09	5.37	3.22	14.70	3.49	5.58
50/50	14.54	3.86	3.90	12.27	2.63	5.90
75/25	12.98	3.61	4.40	12.70	2.73	5.38
90/10	13.49	4.25	4.52	13.59	5.45	5.82

Table 3: Values were calculated using the *split_loop_L* and *split_loop_EN* functions.

We can see that when a small amount of the data is dedicated to training EN performs better than LASSO - it also has a lower st. error. Both perform worse than in the default case of a 50/50 split, however, and LASSO sees a major gain when going from 25/75 to 50/50. As we dedicate more of the data to training, rather than validation, the models begin to agree with each other. The optimal train/validation split is between 50/50 and 90/10 for LASSO - as this is where it achieved its lowest test MSE and lowest MSE st. error. In real-world situations it is common to see a 75/25 or 80/20 split so this result was expected.

Changing number of dataset simulations

The results for both models did not substantially change as we increased the number of datasets (each of the default size of 200). Both LASSO and EN did experience their lowest MSE at the 100 level but not a large improvement over the default.

	Lasso			Elastic Net		
# datasets	MSE	MSE st. error	# vars	MSE	MSE st error	# vars
10	13.83	3.70	4.80	14.05	3.02	6.00
50	14.15	4.14	4.24	13.31	3.07	5.88
100	13.72	4.00	4.00	13.00	3.08	5.90
200	14.57	4.12	3.99	13.32	3.04	5.76
400	14.32	4.17	4.21	13.36	3.25	5.76

Table 4: Values were calculated using the *dataset_loop_L* and *dataset_loop_EN* functions.

Changing the sparsity of the β coefficients

Reducing the number of non-zero β's increased the calculation requirements of both models. It naturally increased the number of variables that could have an effect on the partial residuals. Hence both models suffered as we increased the number of non-zero β's (shown by moving down Table). LASSO performs particularly better when there is a higher number of zero β. Due to its propensity to shrink coefficients to 0 more easily, such a thing would be expected. Because the Elastic Net includes the same penalization method as LASSO, its results are similar in settings with a high sparsity of coefficients. As we increase the number of non-zero β's EN, proves a better prediction model because its error suffers less. It also seems to become very confident on it's estimation on the number of those β, as shown by the decreasing # *vars st. error.*

	Lasso				Elastic Net			
No of $\beta = 0$	MSE	MSE st. error	# vars	# vars st. error	MSE	MSE st. error	# vars	# vars st. error
6	12.37	3.11	4.20	0.70	12.40	2.82	5.00	1.71
5	14.94	4.19	3.52	1.80	12.70	2.92	5.80	1.58
4	17.78	6.89	4.82	1.55	15.92	4.25	6.74	1.35
3	17.99	5.31	4.88	1.85	15.56	4.30	6.90	0.89
2	29.96	13.51	4.88	1.97	18.53	5.36	7.44	0.76
1	27.06	14.25	5.84	1.65	20.67	6.22	7.66	0.59

Table 5: Values were calculated using the *beta_L* and *beta_EN* functions.

Cases when p > n

We generated 3 sets of 20 β coefficients, each one randomly sampled, from 0 to 9, all of them integers:

- $\beta_{set_1} = (2, 9, 9, 7, 6, 0, 0, 3, 3, 2, 0, 7, 4, 3, 7, 1, 7, 0, 1, 2)$
- $\beta_{set_2} = (7, 0, 7, 0, 2, 7, 1, 9, 2, 9, 8, 4, 9, 5, 8, 4, 8, 9, 4, 1)$
- $\beta_{set_3} = (4, 3, 6, 6, 0, 0, 8, 3, 2, 1, 6, 7, 1, 8, 7, 0, 6, 8, 6, 7)$

We tested the models in a setting where there are more predictors than data entries. LASSO struggled to calculate a meaningful MSE and become saturated quickly as the number of β exceeded the number of observations, and EN was naturally worse when $p > n$, but was still able to compute a result.

	Lasso				Elastic Net			
β_{set}	MSE	MSE st. error	# vars	# vars st. error	MSE	MSE st. error	# vars	# vars st. error
1	4e+303	N/A	12.57	5.96	236.28	86.97	18.46	2.85
2	5.07e+304	N/A	15.66	5.80	370.53	126.84	19.10	1.67
3	5.86e+303	N/A	14.55	6.08	306.74	101.81	18.96	1.80

Table 6: Values were calculated using the *beta_L* and *beta_EN* functions.

Although the results for the Elastic Net would not be useful with these results, we found it worthwhile to compare the results to a Ridge Regression, as it uses the same penalizer λ_2 as the Elastic Net. Running the tests with the same 3 sets of β-coefficients, we see that the results are very similar. If anything, the Elastic Net has a somewhat lower standard deviation in its Test MSEs, and tends to have a slightly lower error overall. It is also worth noting that Ridge produces a model with 20 coefficients - it doesn't shrink any to zero. The Elastic net stays in the middle ground between Ridge and LASSO in this aspect. It does perform a more aggressive variable shrinkage than Ridge by bringing some of the coefficients to 0. LASSO here comes clearly short as it cannot produce more coefficient estimates than the amount of entries n for the training data. Ridge is also not optimal, since it produces 20 estimates even if in the original ones we have some that equal 0.

	Ridge			
Trial	MSE	MSE st. error	# vars	# vars st. error
1	238.03	108.97	20	0
2	392.21	157.15	20	0
3	302.26	116.48	20	0

Table 7: Values were calculated using the *beta_R* function.

These last two examples help clearly illustrate the advantages of the Elastic Net. In a setting with high sparsity, it delivers results similar to LASSO, which is the better performer in these settings compared to Ridge. In a setting like the one we just saw where Ridge is better than LASSO, the Elastic Net's results are very similar to those of Ridge. By combining the penalization methods that each uses, one or the other takes more influence depending on which is more useful for the given data.

Study Limitations

When changing one of the parameters, n, train/validation size, and σ, we kept the other parameters at their default values. This is a limitation of our study as it is possible that other combinations had lower average test MSE's due to the interactions between parameters. However, this would have lead to an exponential number of computations.

$$\sigma^{n^{lambda1^{lambda2^{t/v \text{ split}}}}}$$

Where:

- σ denotes the number of sigma values tested
- n denotes the number of n values tested
- $lambda1$ denotes the number of lambda1 values tested
- $lambda2$ denotes the number of lambda2 values tested
- t/v split denotes the number of splits

ANN for Test MSE

For an extension of this project we could have created an Artificial Neural Network (ANN) to regress the parameters we just dealt with (number of datasets, size of training and validation sets, train and validation split, sigma, and the different settings for the coefficients) on $test\_MSE$. This would have allowed us to try further combinations of parameters and explore more scenarios to see in which does either model perform better. From here we could calculate the estimated effect of changing any of those parameters to partials of the actual values we used.

Conclusion

Elastic Net deals better with noise in datasets, as we saw that it performs better at both small and large values of σ in absolute MSE terms and st. error of the MSE. EN performs better in the default case than LASSO, and an added benefit is that it does have a lower st. error in the Test MSE. When we changed other parameter, such as the size of the training and validation datasets, the number datasets on which a test was run, or we tried a better train/validation proportion, both models improved their, and generally the superiority of the Elastic net over LASSO held. Both models suffered when $p > n$ but the Elastic Net was still able to run predictions in this setting, while LASSO wasn't. The model trial in a scenario with more predictors than data entries was particularly useful in seeing how the Elastic Net takes advantage of the best characteristics of both LASSO and Ridge, while generally outperforming both in most cases.