ASM Practice

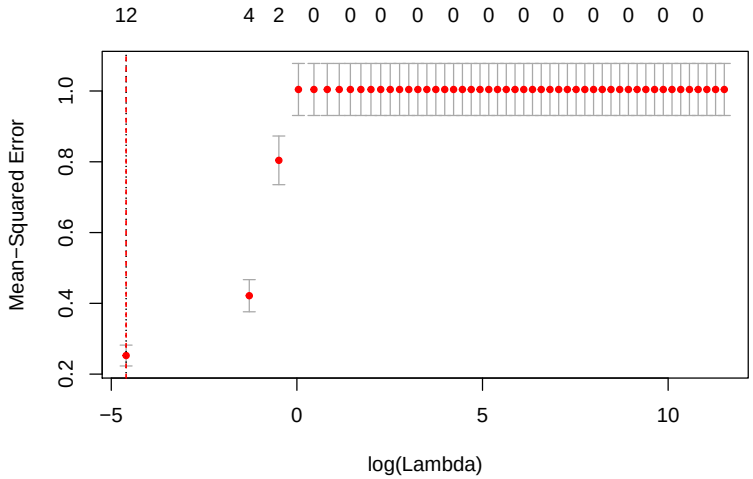
Lasso estimation

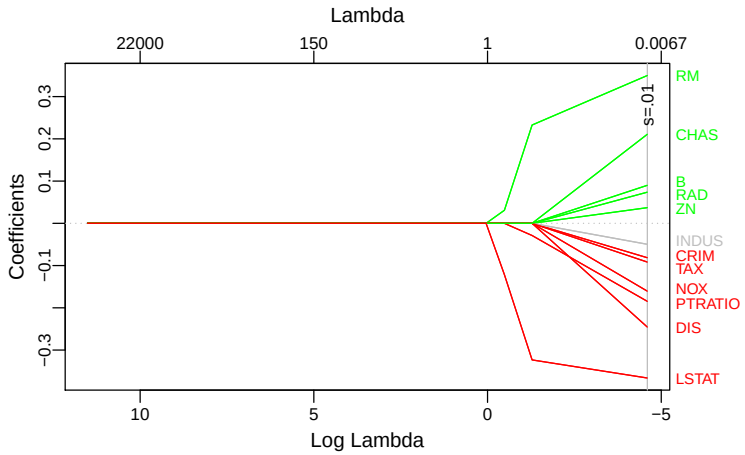
Maria Gkotsopoulou & Ricard Monge Calvo & Amalia Vradi

20/11/2019

Lasso Estimation for the Boston Housing data

We start by scaling and splitting the Boston dataset to training and test using a 2/3 ratio. Since *CHAR* is a factor variable we do not include it in the *scale* function. First we need to tune the parameter λ. To do this we use 10 fold cross validation performed by *cv.glmnet*.





To select the best model, we now use 10-CV using the lambda that best minimised the error in cross-validation, which is 0.0100502.

So our final model has Df= 12 which is the number of non-zero coefficients and $\%Dev$= 0.7755363 is the percent deviance explained, which is quite good.

In terms of interpreting the coefficients, we observe that each additional room (RM) is associated with an increase in the house price, on average. This is quite straightforward, in principle, since it is to be expected

that the larger the house, loosely speaking, the more expensive it will be.

Moreover, an increase in *CHAS* would mean that it will take the value of 1 is associate with an increase in *CMEDV*, so ultimately if the Charles River passes through this suburb then this signals a higher house price, on average.

On the other hand, an increase in *LSTAT* (% lower status of the population) is associated with a decrease in the house price, on average.

Most interestingly though is that the increase in *PTRATIO* (pupil-teacher ratio by town) is associated with a decrease in the house price, on average. So in other words, the education offering of a town increases its value. Also, an increase in *DIS* (weighted distances to five Boston employment centres) is associated with a decrease in *CMEDV*, on average. So, having to do a larger commute to work signals a lower house price.

Another reasonable result is the fact that an increase in *NOX* (nitric oxides concentration) is associated with a decrease in *CMEDV*, so air pollution is a detractor to house price.

Furthermore, we see that *AGE* (proportion of owner-occupied units built prior to 1940) does not seem to have any effect on *CMEDV*, since the coefficient is 0. This was to be expected due to the sparsity of the method.

Finally, we obtain the train and test error.

Table 1: Model Errors Summary

regression_method	train_MSE	test_MSE
lasso estimation	0.224562	0.355448

The difference between train and test errors is not that large, even if the test set is relatively small, and thus subject to a great variance.

Ridge regression comparison

We want to compare the ridge regression models for the same dataset using both the function in the *glmnet* package and our *custom* function. In both cases, we will use 10-fold CV to tune the *lambda* penalization parameter on the training set, build the final model and analyze its performance on the test set.

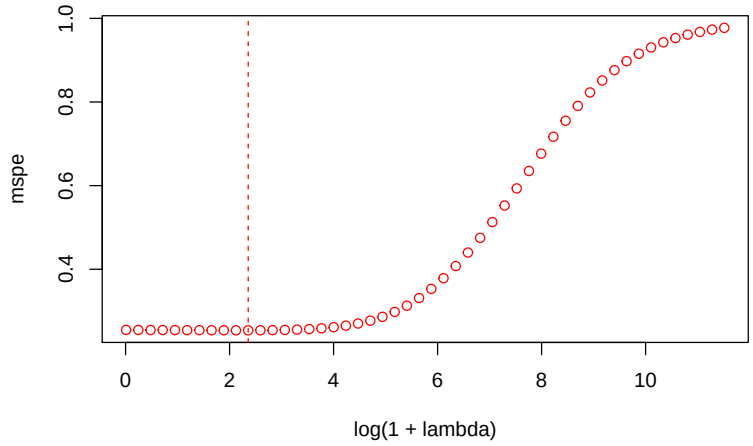
The following table shows the MSPE measures of both methods and for both training and test data:

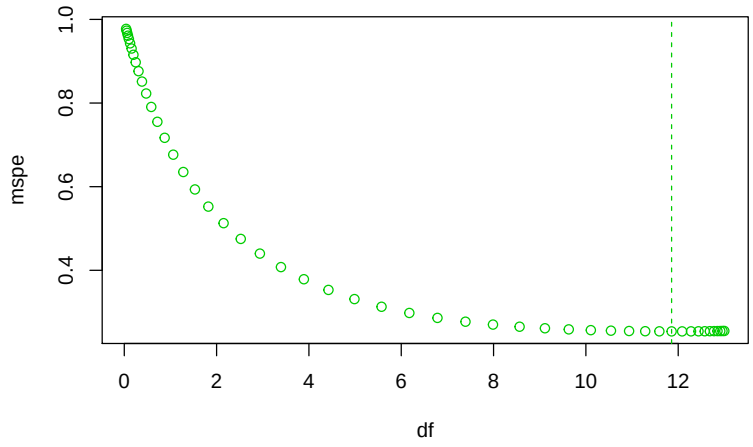
Table 2: Model Errors Summary

regression_method	lambda	train_MSE	test_MSE
Glmnet Ridge	0.010050	0.224562	0.355448
Custom Ridge	9.564907	0.223727	0.356428

We can clearly see the measures are almost the same for both *Glmnet* and *custom* models, and also very similar to the lasso measures.

On the other hand, we see the lambda values for *Glmnet* ridge and lasso are the same while for our custom functions the value is quite different. This could be due to noise or numerical errors in our custom function. To illustrate this, we can plot the $MSPE$ versus $log(1+\lambda)$:





As the curve is almost constant in the lower range of the plot, any small errors could easily affect the minimum without impacting the performance of the model.