Analysis

Liang Hao, Andrew Shibata

10/24/2016

4: Analysis

In the section, we will perform the analysis we discussed above. Specifically, we would perform training and test set creatation, randomization seed establishment, parameter choosing via cross-validation and model fitting for each of the methods.

Pre-modeling Data Processing

We will first perform two major processing steps before the modeling process:

- convert factors into dummy variables

  By transforming the categorizal variables into dummy variables, we may be able to use R's built-in function *glmnet* to fit the data.

- mean centering and standardization

  By centering and Standardizing the data, we may have comparable scales so that the coefficients would function properly.

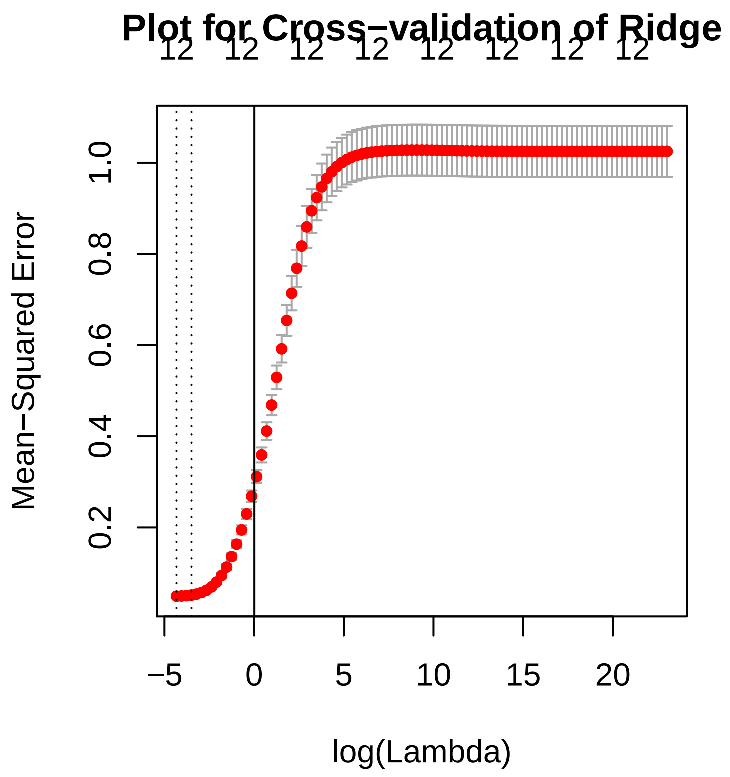
Ordinary Least Squares Regression (OLS)

```
## [1] 0.04749383
```

We first fit the Ordinary Least Squares Regression model to the training set. There is no parameters to be tuned. And the prediction mean square error of the OLS model on the test set is shown above: 0.04749383. We will use this as reference in comparison with other prediction methods.

Ridge Regression (RR)

We then fit the Ridge Regression model to the training set. Firstly, we need to train the model for the best parameter λ via cross-validation. Setting the randomization seed to be 1, we let the `cv.glmnet` function runs with $\lambda = 10^{10}, 10^{9.88}, \ldots, 10^{-2}$.



```
## [1] 0.01321941
```

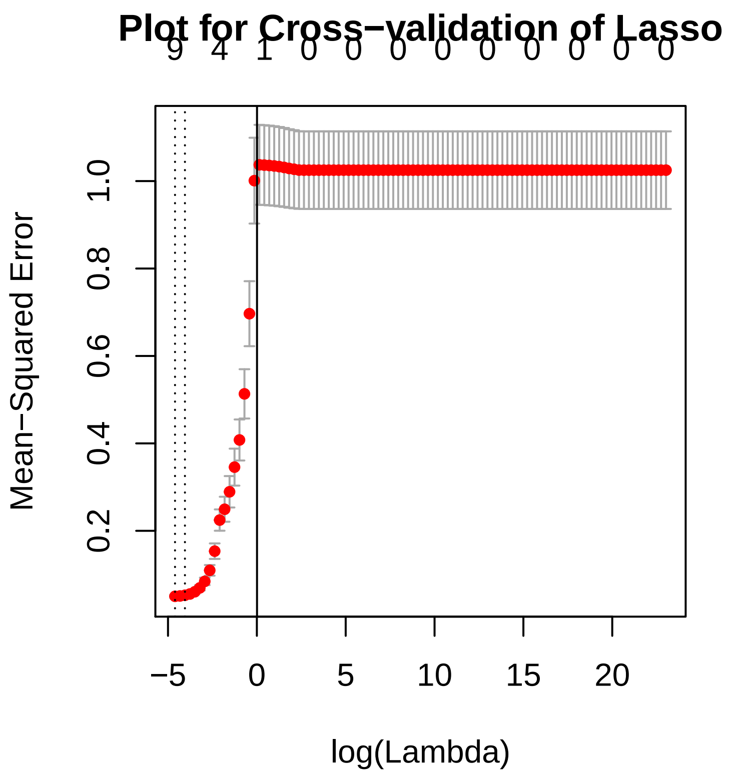
From the graph above, we choose the best λ given this traning set, which is 0.01321941, and refit the model to the entire training set with $\lambda = 0.01321941$.

```
## [1] 0.04611172
```

The resulting mean square error for the same test set is given above. The 0.04611172 does show an improvment compared to the OLS Regression's 0.04749383.. We also fit the model with the best choice of λ to the entire data set for future purpose.

Lasso Regression (LR)

We then fit the Lasso Regression model to the training set. Firstly, we need to train the model for the best parameter λ via cross-validation. Setting the randomization seed to be 1, we let the `cv.glmnet` function runs with $\lambda = 10^{10}, 10^{9.88}, \ldots, 10^{-2}$.



```
## [1] 0.01
```

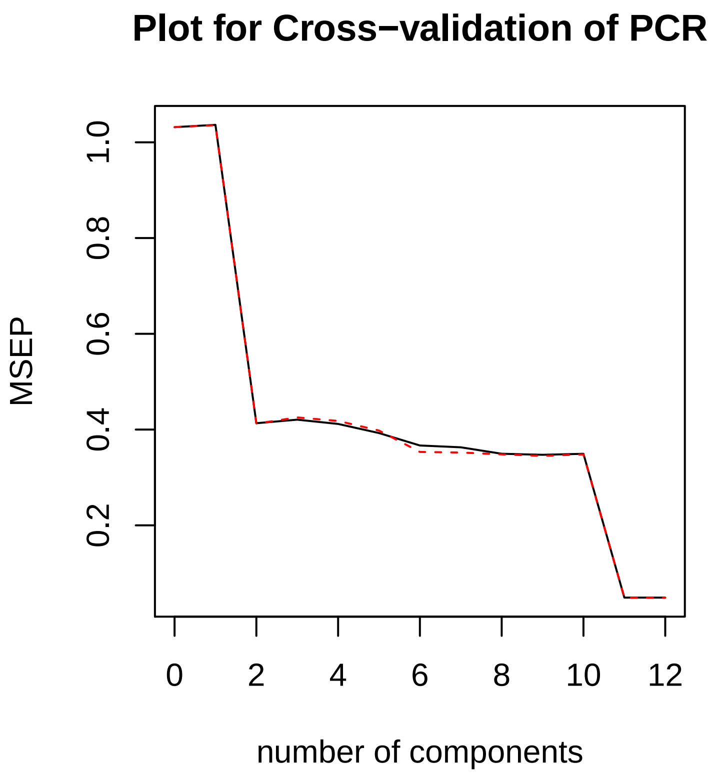
From the graph above, we choose the best λ given this traning set, which is 0.01, and refit the model to the entire training set with $\lambda = 0.01$.

```
## [1] 0.0472898
```

The resulting mean square error for the same test set is given above. The 0.0472898 does show an improvment compared to the OLS Regression's 0.04749383.. We also fit the model with the best choice of λ to the entire data set for future purpose.

Principal Components Regression (PCR)

We then fit the Principal Components Regression to the training set. Setting the randomization seed to be 1, we let the `pcr` function run the cross-validation for the best parameter.



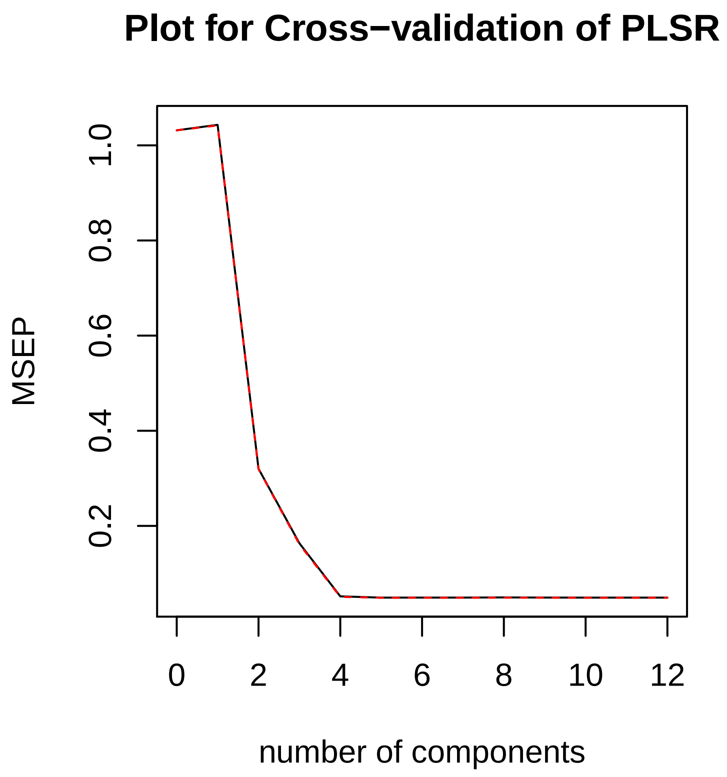
From the validation plot above, we may find that the lowest cross-validation error occurs when $M = 12$ component are used. Therefore, we compute the test MSE:

```
## [1] 0.04749383
```

The resulting mean square error for the same test set is given above. The 0.04749383 is the same as the OLS Regression's 0.04749383. We also fit the model with the best choice of M, the number of components to the entire data set for future purpose.

Partial Least Squares Regression (PLSR)

We then fit the Partial Least Squares Regression to the training set. Setting the randomization seed to be 1, we let the `plsr` function run the cross-validation for the best parameter.



From the validation plot above, we may find that the lowest cross-validation error occurs when $M = 4$ component are used. Therefore, we compute the test MSE:

```
## [1] 0.0465487
```

The resulting mean square error for the same test set is given above. The 0.0465487 does show improvement compared to the OLS Regression's 0.04749383. We also fit the model with the best choice of M, the number of components to the entire data set for future purpose.