Chengjun_Liu_7116099552_hw3

April 13, 2020

1. Did you review up to and including lecture 6 and tutorial 6? If you haven't, please do so first. This homework, like every other homework, only covers a small part of the course contents.

Yes, I reviewed the lectures and tutorials up to 6.

2. Are the following statements correct? (1) Ridge regression tends to give sparser models compared to LASSO. (2) Backward stepwise selection can be applied to situations in which the number of predictors is more than the sample size.

(1) Not correct. On the contrary, LASSO regression tends to give sparser models compared to Ridge regression. Due to the property of L1 Regularization that corners of the parameter contour in the optimization representing more zero parameters tend to contact maximization contours, the ability of LASSO to shrink more features to zero for sparser models is stronger than that of Ridge, which can not fully diminish features due to L2 Regularization.

(2) Not correct. On the contrary, it is forward stepwise selection can be applied to situations in which the number of predictors is more than the sample size. It is because for sparse datasets, the set of significant features tends to be smaller, so forward stepwise selection, which starts from zero features and adds features stepwise, can fit the best model from smaller feature sizes and quickly reach a smaller size of sparse features. While for backward selection, it requires that the number of samples n is larger thanthe number of variables p to fit full models, so it's not suitable for more predictors than sample size.

3. Explain why best subset selection is a computationally intensive method

The best subset selection requires that full combinations of features are all tested to find the best models so that we don't miss any possible better combinations. However, the total feature combinations are $\sum_1^p \binom{p}{i} = 2^n - 1$. As total features increase, the computation increases exponentially ($O(2^n)$). So it is intensive for a larger set of features.

4. What is the imbalance ratio in training.csv? Train Logisitic Regression method with the training dataset in the file training.csv. Then on the test dataset in the file test.csv, calculate the classification error using 1/2 as the threshold of the trained sigmoid function.

[15]:
```
import pandas as pd
import statsmodels.formula.api as smf
```

[3]:
```
training=pd.read_csv('training.csv')
training.head()
```

[3]:

	Index	V1	V2	V3	train_y
0	1	0.735768	1.182321	0.593134	0
1	2	0.926873	1.413204	2.713978	0
2	3	1.127243	0.331993	-1.135993	0
3	4	-0.665470	0.663649	-0.463150	0
4	5	-0.790937	1.149649	-0.138268	0

[12]:
```
testing=pd.read_csv('test.csv')
testing.head()
```

[12]:

	Index	V1	V2	V3	test_y
0	1	1.203008	0.471972	-0.120914	0
1	2	-0.351310	-0.884360	1.427213	0
2	3	1.459216	0.366368	-0.955240	0
3	4	0.429113	-0.495070	0.098336	0
4	5	-1.478627	0.010730	0.294894	0

[8]:
```
IR=sum(training['train_y']==1)/sum(training['train_y']==0)
IR
```

[8]: 10.0

[11]:
```
result=smf.logit('train_y~V1+V2+V3',data=training).fit()
```

```
Optimization terminated successfully.
         Current function value: 0.145483
         Iterations 9
```

[14]:
```
test_prob=result.predict(testing.drop(['Index','test_y'],axis=1))
test_pred=(test_prob>0.5)
y_test=testing['test_y']
classification_error=np.mean(test_pred != y_test)
classification_error
```

[14]: 0.046181818181818185

The imbalance ratio in training.csv is 10, and the classification error for test.csv is 0.046181818181818185.

5. After applying random undersampling to the training dataset, we obtained a new training dataset in the file new_training.csv. What is the imbalance ratio in new_training.csv? Train Logisitic Regression method with the training dataset in the file new_training.csv. Then on the test dataset in the file test.csv (the same test set as you used in problem 4), calculate the classification error using 1/2 as the threshold of the trained sigmoid function.

[16]:
```
new_training=pd.read_csv('new_training.csv')
new_training.head()
```

[16]:

	Index	V1	V2	V3	y
0	1	0.735768	1.182321	0.593134	0
1	2	0.926873	1.413204	2.713978	0
2	3	1.127243	0.331993	-1.135993	0
3	4	-0.665470	0.663649	-0.463150	0
4	5	-0.790937	1.149649	-0.138268	0

[18]:
```
IR=sum(new_training['y']==1)/sum(new_training['y']==0)
IR
```

[18]: 1.0

[19]:
```
result_new=smf.logit('y~V1+V2+V3',data=new_training).fit()
```

```
Optimization terminated successfully.
         Current function value: 0.361158
         Iterations 7
```

[20]:
```
test_prob_new=result_new.predict(testing.drop(['Index','test_y'],axis=1))
test_pred_new=(test_prob_new>0.5)
classification_error_new=np.mean(test_pred_new != y_test)
classification_error_new
```

[20]: 0.10890909090909091

The imbalance ratio in training.csv is 1, and the classification error for test.csv is 0.10890909090909091.

[ ]: