Regularization of Logistic Regression Using Lasso Penalty

J Kou

1. Objective

Logistic regression is the method for predicting a binary outcome with one or more predictionary variables. Overfitting may occur when the model that not only captures the signal, but also the noise in the dataset. Thus, after logistic regression, it is better to do some regularization to avoid overfitting. Generally, there are two methods to do regularization which are Ridge and Lasso. Among them Ridge works by imposing penalty on the magnitude of coefficients, in addition to modify size of coefficients, Lasso can also perform model selection by making coefficients of insignificant variables as zero.

In order to trace the effect of regularization. This project using credit card application training example, estimates the main effect of logistic regression and Lasso regression. The binary outcome variable is application results. The covariates consist of fifteen variables which are replaced by dummy names. Both models are developed using training dataset and applied to the test dataset to evaluate various performance metrics including sensitivity, specificity, and accuracy.

2.Data

2.1 Data source

This data set contains two parts: training data set was used for model selection, after model selection, test data set was used to test the model's prediction accuracy. The original data was downloaded from https://archive.ics.uci.edu/ml/datasets/Credit+Approval. This dataset was used to build a model to predict whether a credit card application can be approved. The names of each predictionary variable was removed from the database, replaced with dummy values.

2.2 Variables

There are 16 variables in the dataset, "Y" refers to the results of application, where "1" indicates approved and "0" indicates denied. Others are the predictors to be used in this project, six of them are continuous variables which are V2, V3, V8, V11, V14, V15; and nine are categorical variables. The information of categorical variables is listed in table 1.

Table 1. information of categorical variables

Y		V1		V5		V6		V7		V9		V10		V12		V13	
level	**n**	**level**	**n**	**level**	**n**	**level**	**n**	**level**	**n**	**level**	**n**	**level**	**n**	**level**	**n**	**level**	**n**
0	299	a	166	g	424	c	110	v	317	f	248	f	297	f	289	g	497
1	248	b	381	gg	1	q	68	h	120	t	299	t	250	t	258	p	2
				p	122	w	50	bb	43							s	48
						i	44	ff	43								
						aa	41	J	7								
						ff	41	Z	7								
						(other)	193	(other)	10								

Note that variables V6 and V7 have more than six levels and some levels contain few observations which will cause overfitting in the prediction. Since the information of them is unknown, it is impossible to combine levels together to lower the number of levels. Thus, variables V6 and V7 are not included in the following model selection. Similarly, V5 only has 1 observation in level “gg”, and there is only “2” observation in level “p” of V13, thus observations containing level “gg” and level “p” were removed in the dataset. After data cleaning, the information of the dataset was visualized in figure 1 and figure 2 which contain information of categorical and continuous variables respectively.

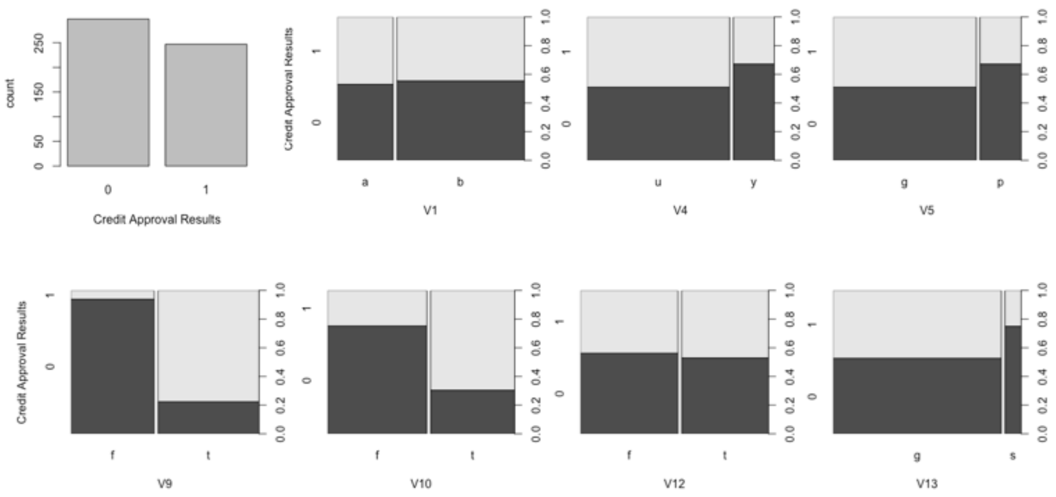

Figure 1. plot of categorical variables V.S. Y

As shown in figure 1, the count of approved and denied observations are almost the same. After data cleaning, there are no categorical variables containing extreme low counts in some levels. Which means these categorical variables are ready for fitting logistic models.

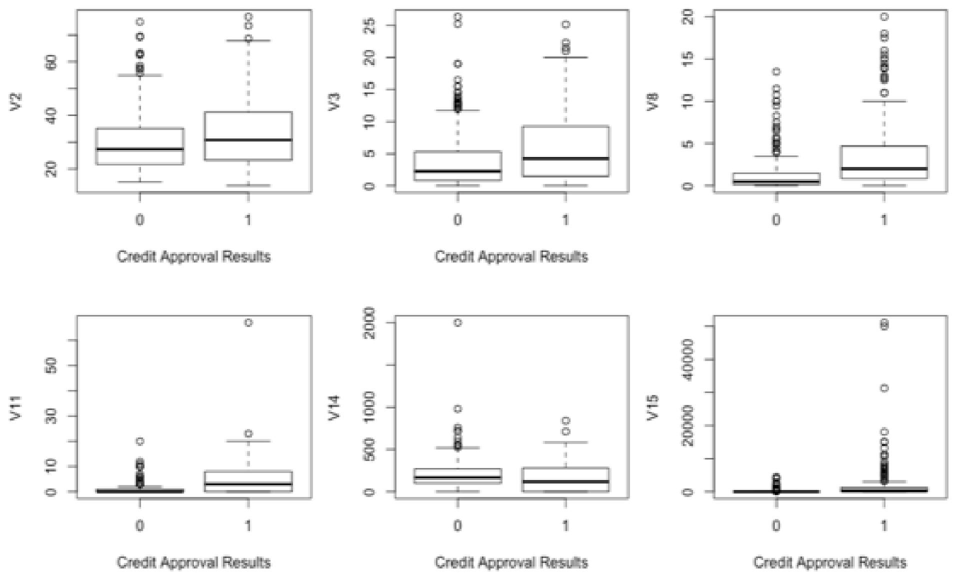

Figure 2. Plot of Continuous variables V.S. Y

Figure 2 shows the relationship between each continuous variable and the results of applications. Most continuous variables have balanced proportions in the approval and deny group. However, V16 has majority values gathered in zero with large numbers of outliers in the approval group (credit approval=1), it will add unnecessary noise to the model prediction, thus V16 was not included in the model selection.

3. Model selection

Forward method was performed in logistic regression. To be more specific, the procedure began with no candidate variables in the model. Select the variable that has the smallest AIC (Akaike Information Criterion). At each step, select the candidate variables that decrease AIC the most. Stop adding variables when none of the remaining variables are significant.

For Lasso regression, the tuning parameter and cut-off point for the Lasso method was estimated using leave-one-out applied to the training dataset. And the estimated tuning parameter in the Lasso model is 0.0094.

Table 2 presents the information of coefficients estimated by logistic regression and Lasso regression.

Table 2. Coefficients of Logistic regression and Lasso regression

variable	**Logistic regression**		**Lasso regression**
	coefficient	Standard error	coefficient
Intercept	-3.041	0.302	-2.779
V9t	3.411	0.318	3.234
V11	0.121	0.053	0.090
V8	0.097	0.049	0.072
V10t	0.668	0.340	0.655
V4y	-0.576	0.308	-0.389

Table 2 shows that both of the models selected the same predictors in the final model. The absolute value of coefficients of the Lasso model is slightly smaller than those of logistic regression. It is reasonable because the tuning parameter of the Lasso model is 0.0094 which is a small number and only shrinks the model in small scale.

4. Training Performance

The cut-off point of the logistic regression model was determined by plotting the ROC curve. ROC is a graph showing the performance of the model at all cut-off points. This curve plots two parameters: true positive rate verses false positive rate. The best cut-off point was determined at the point when the ROC has the highest prediction accuracy. AUC measures the entire area underneath the ROC curve, it provides an aggregate measure of performance across all classification cut-off points. The graph of ROC was shown in figure 3, the AUC of the curve is 90.01%. And the best cut-off point is 0.556 with corresponding best prediction accuracy 85.50%.

Similarly, the cut off points of Lasso regression was 0.5657 with corresponding maximum prediction accuracy 85.50% which is equal to the logistic regression accuracy. It indicates that the performance of two models on the training dataset is the same.

Figure 3. ROC curve of logistic regression

5. Testing performance

Table 3 shows the results of predicting response using logistic regression and Lasso regression, computed by applying models to the test data. The correctly classified cases fall along the main diagonal of the table, while misclassified cases fall along the off-diagonal.

Table 3. Results of logistic regression and Lasso regression

	Logistic regression Prediction			**Lasso regression Prediction**		
Actual	deny	approval	total	deny	approval	total
deny	49	3	52	51	1	52
approval	3	44	47	1	46	47
Total	52	47	99	52	47	99

As can be seen from table 3, both models correctly predicted the response status for a majority of cases.

Table 4 shows the performance statistics for both of the models. Specifically, table 4 shows that Lasso performed much better than logistic regression: Lasso correctly classified 97.98% of all sample observations in the test data, whereas logistic regression only classified 93.94% of the samples correctly. Table 4 also indicates that the true positive rate is considerably higher for Lasso compared to the logistic regression model (Lasso: 97.88%; logistic: 93.62%), and the true negative rate for the Lasso is also higher than logistic model (Lasso: 98.08%; logistic: 94.23%).

Table 4. Statistics of model accuracy

Statistic	**Logistic regression**	**Lasso regression**

Accuracy (Percentage correctly classified)	93.94%	97.98%
Sensitivity (true positive rate)	93.62%	97.88%
Specificity (true negative rate)	94.23%	98.08%

6. Summary

Regularization does not improve the performance on the training dataset. However, it can improve the performance on unseen data, and which is exactly what we want. We don't want the model to memorize the results of what we already know, we want the model to predict results on what we don't know. Thus, it is necessary to apply regularization to model after model selection.

Appendix R code

Training data cleaning

```
setwd("~/Desktop")
cred.1=read.table("project.2.data.1.train.txt",sep=",",na.strings="?")
str(cred.1)

# Cleaning the data and removing the observations with missing values.
sum(is.na(cred.1)) #66
apply(is.na(cred.1),2,sum)
id_no_miss=(apply(is.na(cred.1),1,sum)==0)
cred.2=cred.1[id_no_miss,]
sum(is.na(cred.2)) #0
#change dependent variable value to 0 and 1
```

```
Y=rep(0,nrow(cred.2))
Y[cred.2[,16]=="+"]=1
Y[cred.2[,16]=="-"]=0
cred.3=cbind(cred.2[-16],Y)
cred.3$Y=as.factor(cred.3$Y)
str(cred.3)
```

---Table 1---

```
summary(cred.3[,c(1,4,5,6,7,9,10,12,13,16)])
----output------
V1     V4      V5          V6          V7      V9      V10     V12     V13     Y
 a:166   l:  1   g :424   c      :110   v      :317   f:248   f:297   f:289   g:497   0:299
 b:381   u:424   gg:  1   q      : 68   h      :120   t:299   t:250   t:258   p:  2   1:248
         y:122   p :122   w      : 50   bb     : 43                           s: 48
                          i      : 44   ff     : 43
                          aa     : 41   j      :  7
                          ff     : 41   z      :  7
                          (Other):193   (Other): 10  )
----output end-----
```

```
cred.3=cred.3[-row(cred.3)[which(cred.3$V4=="l")],]
cred.3$V4=as.numeric(cred.3$V4)
cred.3$V4[cred.3$V4==2]="u"
cred.3$V4[cred.3$V4==3]="y"
cred.3$V4=as.factor(cred.3$V4)
summary(cred.3$V4)
cred.3$V5=as.numeric(cred.3$V5)
cred.3$V5[cred.3$V5==1]="g"
cred.3$V5[cred.3$V5==3]="p"
cred.3$V5=as.factor(cred.3$V5)
summary(cred.3$V5)
cred.3=cred.3[-row(cred.3)[which(cred.3$V13=="p")],]
cred.3$V13=as.numeric(cred.3$V13)
cred.3$V13[cred.3$V13==1]="g"
cred.3$V13[cred.3$V13==3]="s"
cred.3$V13=as.factor(cred.3$V13)
summary(cred.3$V13)
```

data visualization

---Figure 1---

```
plot(cred.3$Y, xlab="Credit Approval Results", ylab="count")
spineplot(Y~V1,data=cred.3,ylab="Credit Approval Results",xlab="V1")
spineplot(Y~V4,data=cred.3,ylab="Credit Approval Results",xlab="V4")
spineplot(Y~V5,data=cred.3,ylab="Credit Approval Results",xlab="V5")
spineplot(Y~V6,data=cred.3,ylab="Credit Approval Results",xlab="V6")
spineplot(Y~V7,data=cred.3,ylab="Credit Approval Results",xlab="V7")
spineplot(Y~V9,data=cred.3,ylab="Credit Approval Results",xlab="V9")
spineplot(Y~V10,data=cred.3,ylab="Credit Approval Results",xlab="V10")
spineplot(Y~V12,data=cred.3,ylab="Credit Approval Results",xlab="V12")
spineplot(Y~V13,data=cred.3,ylab="Credit Approval Results",xlab="V13")
```

---Figure 2--

```
boxplot(V2~Y,data=cred.3,xlab="Credit Approval Results",ylab="V2")
boxplot(V3~Y,data=cred.3,xlab="Credit Approval Results",ylab="V3")
boxplot(V8~Y,data=cred.3,xlab="Credit Approval Results",ylab="V8")
boxplot(V11~Y,data=cred.3,xlab="Credit Approval Results",ylab="V11")
boxplot(V14~Y,data=cred.3,xlab="Credit Approval Results",ylab="V14")
boxplot(V15~Y,data=cred.3,xlab="Credit Approval Results",ylab="V15")
#data for analysis
cred.4=cred.3
cred=cred.4[-c(6,7,13,15)]
str(cred)
```

fit model without regularity

```
fit.null=glm(Y~1,family=binomial, data=cred)
fit.full=glm(Y~.,family=binomial, data=cred)
select.1=step(fit.null, scope=list(lower=fit.null,upper=fit.full),direction="forward")
fit.1=glm(Y ~ V9 + V11 + V8 + V10 + V4, family = binomial, data = cred)
```

---Table 2--

```
summary(fit.1)
 ---output----
Call:
glm(formula = Y ~ V9 + V11 + V8 + V10 + V4, family = binomial,
   data = cred)

Deviance Residuals:
   Min       1Q   Median       3Q      Max
-2.6397  -0.3416  -0.2330   0.5721   2.6995

Coefficients:
          Estimate Std. Error z value Pr(>|z|)
(Intercept) -3.04126    0.30180 -10.077   <2e-16 ***
V9t          3.41104    0.31786  10.731   <2e-16 ***
V11          0.12074    0.05289   2.283   0.0224 *
V8           0.09677    0.04870   1.987   0.0469 *
V10t         0.66818    0.34000   1.965   0.0494 *
V4y         -0.57598    0.30810  -1.869   0.0616 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

   Null deviance: 750.75  on 544  degrees of freedom
Residual deviance: 388.10  on 539  degrees of freedom
```

```
AIC: 400.1

Number of Fisher Scoring iterations: 5
-----output end-----
```

ROC

```
pi0=seq(1,0, length.out = 100)
nsample=nrow(cred)
roc.1=NULL
cv.set.list=split(sample(nsample),rep(1:10,length=nsample))  #10-fold cross-validation method
for(k in 1:length(pi0))
{n_11.1=n_10.1=n_01.1=n_00.1=0
nsample=nrow(cred)
for(i in 1:length(cv.set.list))
{
  training=cred[-cv.set.list[[i]],]
  test=cred[cv.set.list[[i]],]
  fit.1.training=glm(Y~V9 + V11 + V8 + V10 + V4, data=training, family="binomial")
  for(j in 1:length(cv.set.list[[i]]))
  {if(predict(fit.1.training, test[j,], type="response")>=pi0[k])
  {Y.pred.1=1}else{Y.pred.1=0}
    if((test$Y[j]==1)&(Y.pred.1==1))
    {n_11.1=n_11.1+1}
    if((test$Y[j]==1)&(Y.pred.1==0))
    {n_10.1=n_10.1+1}
    if((test$Y[j]==0)&(Y.pred.1==1))
    {n_01.1=n_01.1+1}
    if((test$Y[j]==0)&(Y.pred.1==0))
    {n_00.1=n_00.1+1}
  }
}
sen=n_11.1/(n_10.1+n_11.1)
spe=n_00.1/(n_00.1+n_01.1)
roc.1=rbind(roc.1, c(1-spe, sen))
}
```

---Figure 3---

```
plot(roc.1, type="s",xlim=c(0,1), ylim=c(0,1), col="red", main="ROC curve", xlab="1-Specificity",
ylab="Sensitivity")
auc.1=sum(roc.1[-100, 2]*(roc.1[-1,1]-roc.1[-100,1]))
auc.1 #AUC fit.1 0.9001304

#prediction accuracy
pred.accuracy.1=roc.1[,2]*sum(cred$Y==1)/nsample+(1-roc.1[,1])*sum(cred$Y==0)/nsample
z.1=which.max(pred.accuracy.1)
pi0[z.1] #best cut-off points of fit.1 0.5555556
pred.accuracy.1[z.1]  #best prediction accuracy for fit.1 0.8550459
```

testing

```
#testing data
cred.test=read.table("project.2.data.2.test.txt",sep=",",na.strings="?")
sum(is.na(cred.test)) #0
str(cred.test)
summary(cred.test$V4)
cred.test=cred.test[-row(cred.test)[which(cred.test$V4=="l")],]
cred.test$V4=as.numeric(cred.test$V4)
cred.test$V4[cred.test$V4==2]="u"
cred.test$V4[cred.test$V4==3]="y"
cred.test$V4=as.factor(cred.test$V4)
Y=rep(0,nrow(cred.test))
Y[cred.test[,16]=="+"]=1
Y[cred.test[,16]=="-"]=0
cred.test.2=cbind(cred.test[-16],Y)
cred.test.2$Y=as.factor(cred.test.2$Y)
```

---Table 3---

```
test=cred.test.2
Y.pred.test.1=predict(fit.1,test, type="response")
Y.pred.test1=rep(0,nrow(test))
Y.pred.test1[Y.pred.test.1>=pi0[z.1]]=1
n_11.1=sum((test$Y==1)&(Y.pred.test1==1))
n_11.1 # 44
n_10.1=sum((test$Y==1)&(Y.pred.test1==0))
n_10.1 #3
n_01.1=sum((test$Y==0)&(Y.pred.test1==1))
n_01.1 #3
n_00.1=sum((test$Y==0)&(Y.pred.test1==0))
n_00.1 #49
```

---Table 4---

```
#sen spe acc
sen.test1=n_11.1/(n_10.1+n_11.1)
spe.test1=n_00.1/(n_00.1+n_01.1)
acc.test1=(n_11.1+n_00.1)/nrow(test)
sen.test1 #test sensitivity fit.1 0.9361702
spe.test1 # test specificity fit.1 0.9423077
acc.test1 #test prediction accuracy fit.1 0.9393939
```

Lasso regression

```
y=as.factor(cred$Y)
x=model.matrix(fit.full)[,-1]
library(foreach)
library(glmnet)
```

```
fit.4=glmnet(x,y,family="binomial")
lambda.seq=fit.4$lambda
pi0=seq(0,1,length.out=100)
correct.num=matrix(0,length(lambda.seq),length(pi0))
nsample=nrow(cred)
for(i in 1:nsample) {
  x.train=x[-i,]
  x.test=matrix(x[i,],1,ncol(x))
  y.train=y[-i]
  y.test=y[i]
  fit.1.training=glmnet(x.train, y.train, family="binomial")
  for(j in 1:length(lambda.seq)) {
    pred.prob=predict(fit.1.training, newx=x.test, s=lambda.seq[j], type="response")
    for(k in 1:length(pi0)) {
      if(pred.prob>=pi0[k]) { Y.pred.1=1}else {Y.pred.1=0}
      if((y.test==1)&(Y.pred.1==1))
      {correct.num[j,k]=correct.num[j,k]+1}
      if((y.test==0)&(Y.pred.1==0))
      {correct.num[j,k]=correct.num[j,k]+1} }
  }
}
accuracy=correct.num/nsample
max(accuracy)
matrix=which(accuracy==max(accuracy), arr.ind=TRUE)
j=matrix[1,1]
k=matrix[1,2]
lambda.seq[j] # tuning parameter 0.009449769
pi0[k] #cut to cut off point 0.5656566
#fit Lasso model
lambda.opt=lambda.seq[j]
fit.lasso=glmnet(x,y,family="binomial")
```

---Table 2--

```
coef(fit.lasso, s=lambda.opt)
 ----output-----
12 x 1 sparse Matrix of class "dgCMatrix"
                       1
(Intercept) -2.778877e+00
V1b          .
V2           .
V3           .
V4y        -3.885566e-01
V5p        -2.529868e-15
V8          7.194622e-02
V9t         3.233533e+00
V10t        6.545487e-01
V11         8.993386e-02
V12t         .
V14        -2.590478e-04
----output end-----
```

---Table 3---

```
#test accuracy
test=cred.test.2
x.test=data.matrix(test[,-16])
y.test=test[,16]
fit.lasso.test=glmnet(x.test,y.test,family="binomial")
Y.pred.3.1=predict(fit.lasso.test, newx=x.test, s=0.009449769, type="response")
Y.pred.3=rep(0,nrow(test))
Y.pred.3[Y.pred.3.1>0.5656566]=1
Y.pred.3
-------------output-------------------
[1] 0 0 0 0 1 1 1 1 0 0 0 1 0 0 0 0 0 1 1 1 1 0 0 1 1 1 1 1 0 0 1 1 1 1 1 0 0 1 0 0 0 0 0 0 0 0 1 1
[49] 1 0 0 0 0 1 0 0 1 0 0 0 0 0 1 1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 1
[97] 1 0 1
----------------end--------------------

n_11.3=sum((test$Y==1)&(Y.pred.3==1))
n_11.3 #46
n_10.3=sum((test$Y==1)&(Y.pred.3==0))
n_10.3 #1
n_01.3=sum((test$Y==0)&(Y.pred.3==1))
n_01.3 #1
n_00.3=sum((test$Y==0)&(Y.pred.3==0))
n_00.3 #51
```

---Table 4---

```
sen.3=n_11.3/(n_10.3+n_11.3)
spe.3=n_00.3/(n_00.3+n_01.3)
acc.3=(n_11.3+n_00.3)/nrow(test)
sen.3 #test sensitivity lasso 0.9787234
spe.3 # test specificity lasso 0.9807692
acc.3 #test prediction accuracy lasso 0.979798
```