# Comparison Of Two Tests

*Zheng Yuan*

*2019/9/9*

```r
library(simstudy)
library(dplyr)
library(caret)
library(lmtest)
library(knitr)
```

### Define function to evaluate test statistic based on loocv prediction error

Step1: Leave one out cross validation prediction error is evaludated by loocv() function, which is built based on

$$LOOCV = \frac{1}{n}\sum_{i=1}^{n}\frac{(Y_i - \hat{Y}_i)^2}{(1-h_{ii})^2}$$

Step2: Evaluate new test statistic based on

$$New\ Test\ Statistic = (\frac{n}{\sigma^2}\hat{\Gamma}_{\alpha,n} - 2d_\alpha) - (\frac{n}{\sigma^2}\hat{\Gamma}_{\lambda,n} - 2d_\lambda)$$

, where $M_\alpha \subset M_\lambda$.

```r
#### leave one out cross validation prediction error function using trick

loocv=function(fit){

  h=lm.influence(fit)$h
  mean((residuals(fit)/(1-h))^2)

}

newts <- function(model1,model2) {
  (length(model1$model$y) * loocv(model1)
   -2*model1$rank)- (length(model2$model$y) * loocv(model2)-2*model2$rank)
}
```

### Define likehood ratio test function

(1)First step: Evaluate the maximum likelihood estimation of $\sigma^2$, that is

$$\hat{\sigma_\alpha^2} = \frac{1}{n}RSS_\alpha = \frac{1}{n}(Y - X_\alpha\hat{\beta_\alpha})^T(Y - X_\alpha\hat{\beta_\alpha})$$

for any certain model $M_\alpha$. Let's call this funtion **'mse()'**.

(2)Second step: Assuming normality, the maximum log-likelihood value for model $\alpha$, $l(\hat{\beta_\alpha}, \hat{\sigma_\alpha^2})$ say, for the full-rank regresssion model is the log probability density of $Y$ with maximum likelihood estimation as parameters, namely,

$$\begin{aligned} l(\hat{\beta_\alpha},\hat{\sigma_\alpha^2}) &= log\Big\{(2\pi\hat{\sigma_\alpha^2})^{-\frac{n}{2}}exp\Big\{-\frac{1}{2\hat{\sigma_\alpha^2}}||Y-X_\alpha\hat{\beta_\alpha}||^2\Big\}\Big\} \\ &= -\frac{n}{2}log(2\pi\hat{\sigma_\alpha^2})-\frac{1}{2\hat{\sigma_\alpha^2}}||Y-X_\alpha\hat{\beta_\alpha}||^2 \\ &= -\frac{n}{2}log(2\pi)-\frac{n}{2}log\frac{RSS_\alpha}{n}-\frac{n}{2} \\ &= C-\frac{n}{2}logRSS_\alpha \end{aligned}$$

where C does not depend on model $M_alpha$
(3)Third step: Evaluate log likelihood ratio test statistic according to definition, namely,

$$LLR=-2(l(\hat{\beta_\alpha},\hat{\sigma_\alpha^2})-l(\hat{\beta_\lambda},\hat{\sigma_\lambda^2}))=n(logRSS_\alpha-logRSS_\lambda)$$

.

```
mse <- function(object) {
  mean(residuals(object)^2)
}

lrt <- function(model1,model2) {
  length(model1$model$y) * (log(mse(model1)) - log(mse(model2)))
}
```

## Define a full rank dataset and set true model

The true linear model is y=3+2*x1+5*x2+3*x3+1.5*x4+e, which is the full model.

```
#### Define A Full Rank Dataset

def <- defData(varname = "x1", dist="uniform",formula = "10;20")  ## x1 is from unifrom distribution

def <- defData(def,varname = "x2", dist="uniform",formula = "0;3")

def <- defData(def,varname = "x3", dist="uniform",formula = "0;5")

def <- defData(def,varname = "x4", dist="uniform",formula = "5;10")

def <- defData(def, varname = "y", formula = "3+2*x1+5*x2+3*x3+1.5*x4", variance = 1)

## The true linear model is y=3+2*x1+5*x2+3*x3+1.5*x4+e
```

## Generate a dataset with size 1000 and fit each model

Here I wanted to do a single time simulation to see exactly what values the test statstics are under the alternative hypothesis, i.e., under full model. In this simulation, I generated a dataset of size 1000 and fit model 1~4, respectively.

To test the power of a certain test, basically we set the null hypothesis as model 1~3 respectively, because they are all incorrect (insufficient) model compared with the model under alternative hypothsis, which we set as the true model. The goal is to see exactly to what extent can we reject the null hypothesis in each test.

```r
md<-c()

n = 1000

dt <- genData(n, def)

dt <- dt%>%select(y,x1,x2,x3,x4) ##generate dataset n=1000

fit1 <- lm(y~ x1, data = dt)

fit2 <- lm(y ~ x1+x2, data = dt)

fit3 <- lm(y ~ x1+x2+x3, data = dt)

fit4 <- lm(y ~ x1+x2+x3+x4, data = dt)
```

### Compare the corresponding value of two test statistics

```r
ts1 = newts(fit1,fit4)

ts2 = newts(fit2,fit4)

ts3 = newts(fit3,fit4)


lr1 = lrt(fit1,fit4)

lr2 = lrt(fit2,fit4)

lr3 = lrt(fit3,fit4)

ts1;lr1;ts2;lr2;ts3;lr3
```

```
## [1] 42381.89
## [1] 3760.627
## [1] 23856.18
## [1] 3201.892
## [1] 5068.212
## [1] 1791.77
```

Now do the same simulation once again, only with sample size of 5000 this time.

### Generate a dataset with size 5000

```r
md<-c()

n = 5000
```

```r
dt <- genData(n, def)

dt <- dt%>%select(y,x1,x2,x3,x4) ##generate dataset n=1000

fit1 <- lm(y~ x1, data = dt)

fit2 <- lm(y ~ x1+x2, data = dt)

fit3 <- lm(y ~ x1+x2+x3, data = dt)

fit4 <- lm(y ~ x1+x2+x3+x4, data = dt)

md<-c(loocv(fit1),loocv(fit2),loocv(fit3),loocv(fit4)) ##loocv prediction error for each model
```

## Calculate the test statistic in one time simulation

```r
ts1 = newts(fit1,fit4)

ts2 = newts(fit2,fit4)

ts3 = newts(fit3,fit4)

lr1 = lrt(fit1,fit4)

lr2 = lrt(fit2,fit4)

lr3 = lrt(fit3,fit4)

ts1;lr1;ts2;lr2;ts3;lr3
```

```
## [1] 214967.5

## [1] 19048.56

## [1] 119704

## [1] 16206.71

## [1] 22791.19

## [1] 8680.96
```

Interpretation: from the test statistics value above, we can see that in this one-time simulation, our new test statistics are always much greater than their corresponding traditional maximum likelihood ratio test statistics, which implies
that our new test enjoys a much higher power. Besides, the difference becomes larger as the difference between dimensions and sample size n become larger.

## 1000 times simulations

Now let's make 1000 times simulations, in each time to see whether our new test statistic is greater than its counterpart log likelihood ratio test statistic. Since they share the same criterion, a larger value implies a greater power of the test.

```r
df1 <- c()
df2 <- c()
df3 <- c()

for(i in 1:1000){

  dt <- genData(100, def)%>%select(y,x1,x2,x3,x4)
  ##generate dataset n=100
  ##The true linear model is y=3+2*x1+5*x2+3*x3+1.5*x4+e
  fit1 <- lm(y~ x1, data = dt)

  fit2 <- lm(y ~ x1+x2, data = dt)

  fit3 <- lm(y ~ x1+x2+x3, data = dt)

  fit4 <- lm(y ~ x1+x2+x3+x4, data = dt)

  ts1 = newts(fit1,fit4)

  ts2 = newts(fit2,fit4)

  ts3 = newts(fit3,fit4)


  lr1 = lrt(fit1,fit4)

  lr2 = lrt(fit2,fit4)

  lr3 = lrt(fit3,fit4)


 df1 <- c(df1,ifelse (ts1 > lr1, 1, 0 ) )

 df2 <- c(df2,ifelse (ts2 > lr2, 1, 0 ) )

 df3 <- c(df3,ifelse (ts3 > lr3, 1, 0 ) )
## In each simulation record if new test statistics is larger than traditional maximum likelihood ratio

}
```

## Show the result

```r
table(df1)
```

```
## df1
##    1
## 1000
```

```r
table(df2)
```

```
## df2
##    1
## 1000
```

```r
table(df3)
```

```
## df3
##    1
## 1000
```

Interpretation: from the result, we can see that in this 1000 times mulations, our new test statistics are greater than traditional maximum likelihood ratio test statistics each single time, which implies that the new test has the same level as maximum likelihood ratio tes while enjoys a higher power and its performance is very stable. So it can beat likelihood ratio test on power.

## Assume $\sigma^2$ is unknown in new testing procedure

Admittedly, in the comparison above, we assume that $\sigma^2$ is known in new testing procedure while it is unknown in likelihood ratio test. To make the competition more "fair", let us also assume that $\sigma^2$ is unknown in new testing procedure and use the mean of RSS to estimate $\sigma^2$, as we do in likelihood ratio test.

Specifically in this case, instead of assuming $\sigma^2 = 1$, we use the mean of RSS of the larger model to estimate $\sigma^2$, then the test statistic becomes

$$(\frac{n}{\hat{\sigma}^2_\lambda}\hat{\Gamma}_{\alpha,n} - 2d_\alpha) - (\frac{n}{\hat{\sigma}^2_\lambda}\hat{\Gamma}_{\lambda,n} - 2d_\lambda)$$

```r
newts1 <- function(model1,model2) {
  (length(model1$model$y)/mse(model2)* loocv(model1)-2*model1$rank)
  -(length(model2$model$y)/mse(model2) * loocv(model2)-2*model2$rank)
}

n = 5000

dt <- genData(n, def)

dt <- dt%>%select(y,x1,x2,x3,x4) ##generate dataset with sample size n=5000

fit1 <- lm(y~ x1, data = dt)

fit2 <- lm(y ~ x1+x2, data = dt)

fit3 <- lm(y ~ x1+x2+x3, data = dt)

fit4 <- lm(y ~ x1+x2+x3+x4, data = dt)

ts1 = newts1(fit1,fit4)

ts2 = newts1(fit2,fit4)

ts3 = newts1(fit3,fit4)

lr1 = lrt(fit1,fit4)

lr2 = lrt(fit2,fit4)

lr3 = lrt(fit3,fit4)
```

```
ts1;lr1;ts2;lr2;ts3;lr3
```

```
## [1] -4999.958

## [1] 18848.96

## [1] -4999.958

## [1] 15846.93

## [1] -4999.958

## [1] 8608.283
```

we can see in the result that after replacing $\sigma^2$ with its mle estimation under the more complex model, the values of the new test statistics have no big difference.

But what's insteresting is, if we replace $\sigma^2$ with its mle estimation under each model respectively instead of using a universal estimator, say,

$$(\frac{n}{\hat{\sigma_\alpha^2}}\hat{\Gamma}_{\alpha,n} - 2d_\alpha) - (\frac{n}{\hat{\sigma_\lambda^2}}\hat{\Gamma}_{\lambda,n} - 2d_\lambda)$$

, then the test statistic would "mess up".

```
newts1 <- function(model1,model2) {
  (n/mse(model1)* loocv(model1)-2*model1$rank)-(n/mse(model2) * loocv(model2)-2*model2$rank)
}

n = 5000

dt <- genData(n, def)

dt <- dt%>%select(y,x1,x2,x3,x4) ##generate dataset with sample size n=5000

fit1 <- lm(y~ x1, data = dt)

fit2 <- lm(y ~ x1+x2, data = dt)

fit3 <- lm(y ~ x1+x2+x3, data = dt)

fit4 <- lm(y ~ x1+x2+x3+x4, data = dt)

ts1 = newts1(fit1,fit4)

ts2 = newts1(fit2,fit4)

ts3 = newts1(fit3,fit4)

lr1 = lrt(fit1,fit4)

lr2 = lrt(fit2,fit4)

lr3 = lrt(fit3,fit4)

ts1;lr1;ts2;lr2;ts3;lr3
```

```
## [1] -0.2761022
## [1] 18820.11
```

```
## [1] -0.1858791
```

```
## [1] 15925.24
```

```
## [1] -0.1272594
```

```
## [1] 8670.847
```

We can see in this case that the test statistics go to zeroes directly or even less than zero, which is not hard to understand, since $\hat{\sigma_\alpha^2}$ is close to $\hat{\Gamma}_{\alpha,n}$ when n goes to infinity. In fact, using the mle of $sigma^2$ from larger model is already sufficient, cause larger model contains small model.