Sto Simulation Report

Yifan Ma

2022-05-23

Simulation Report on the prediction mean square error (PMSE) calculation

by [**?**] "Predictive mean square error and stochastic regressor variables"

Method

The thesis "Predictive mean square error and stochastic regressor variables" by Narula studied the problem where all of the predictor variables are random, which is stochastic, and follow a multivariate normal distribution. For this problem, he discussed the subset approach in Section 3.1 and gave the expression of unconditional PMSE and conditional PMSE given new data z_0 for full model and subset model.

The response variable and the predictor variables have a joint $(k+1)$-variate normal distribution with unknown mean $\mu^* = [\mu_0, \mu_1, ..., \mu_k]' = [\mu_0, \boldsymbol{\mu'}]'$ and unknown covariate matrix $\boldsymbol{\Sigma^*} = \begin{bmatrix} \sigma_{00} & \boldsymbol{\sigma'} \\ \boldsymbol{\sigma} & \boldsymbol{\Sigma} \end{bmatrix}$.

Let $\boldsymbol{z_1, z_2, ..., z_n}$ be n independent (k-component vector) observations on the predictor variables, $\boldsymbol{x_i = z_i - \bar{z}}$. We assume the correct the model(1.1)

$$\boldsymbol{y = \alpha + \beta_1 z_1 + \beta_2 z_2 + ... + \beta_k z_k + \epsilon}$$

The LSE prediction equation

$$\boldsymbol{\hat{y}} = \bar{y} + \boldsymbol{\hat{\beta}_1}(\boldsymbol{z_1} - \bar{z}_1) + \boldsymbol{\hat{\beta}_2}(\boldsymbol{z_2} - \bar{z}_2) + ... + \boldsymbol{\hat{\beta}_k}(\boldsymbol{z_k} - \bar{z}_k) = \bar{y} + \boldsymbol{X'\hat{\beta}}$$

$$\hat{y}_i = \bar{y} + \boldsymbol{x_i'\hat{\beta}}$$

For any given $\boldsymbol{z_i}$,

$$\begin{aligned} E(y_i|\boldsymbol{z_i}) &= \alpha + \boldsymbol{\beta z_i} \\ &= \mu_0 - \boldsymbol{\sigma}'\boldsymbol{\Sigma^{-1}\mu} + \boldsymbol{\sigma'\Sigma^{-1}z_i} \\ &= \mu_0 + \boldsymbol{\sigma}'\boldsymbol{\Sigma^{-1}(z_i - \mu)} \end{aligned}$$

where $\alpha = \mu_0 - \boldsymbol{\sigma}'\boldsymbol{\Sigma^{-1}\mu}, \boldsymbol{\beta = \sigma'\Sigma^{-1}}$. Thus $\hat{\alpha} = \bar{y} - \boldsymbol{s'S^{-1}\bar{x}}, \boldsymbol{\hat{\beta} = S^{-1}s}$. The conditional predictive mean square error is given by

$$E[(y_0 - \hat{y_0})^2|\boldsymbol{z_0}] = \sigma_k^2\left(1 + \frac{1}{n}\right) + \sigma_k^2\left[\boldsymbol{(z_0 - \mu)'\Sigma^{-1}(z_0 - \mu)} + \frac{k}{n}\right]\frac{1}{n-k-2}$$

and unconditional PMSE by

$$E[(y_0 - \hat{y_0})^2] = \sigma_k^2\left(1 + \frac{1}{n}\right)\left(\frac{n-2}{n-k-2}\right)$$

For subset, we partition the k-component vector of predictor variables into two parts, $\boldsymbol{Z} = [\boldsymbol{Z_1}, \boldsymbol{Z_2}], \boldsymbol{X} = [\boldsymbol{X_1}, \boldsymbol{X_2}], \boldsymbol{x_1'} = [\boldsymbol{x_{i1}'},$ $\boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma_{11}} & \boldsymbol{\Sigma_{12}} \\ \boldsymbol{\Sigma_{21}} & \boldsymbol{\Sigma_{22}} \end{bmatrix}, \boldsymbol{S} = \begin{bmatrix} \boldsymbol{S_{11}} & \boldsymbol{S_{12}} \\ \boldsymbol{S_{21}} & \boldsymbol{S_{22}} \end{bmatrix}$, so the subset prediction equation is given by

$$\tilde{y_i} = \bar{y} + \boldsymbol{x_{i1}'} \tilde{\boldsymbol{\beta_1}}$$

where $\tilde{\boldsymbol{\beta_1}} = \boldsymbol{S_{11}^{-1}} \boldsymbol{s_1}$, where $\boldsymbol{\Phi_1} = \boldsymbol{\beta_1} + \boldsymbol{\Sigma_{11}^{-1}} \boldsymbol{\Sigma_{12}} \boldsymbol{\beta_2}$, the conditional PMSE at z_0 is given by

$$E[(y_0 - \tilde{y_0})^2 | \boldsymbol{z_0}] = \sigma_k^2 + \frac{\sigma_p^2}{n} + \sigma_p^2 \left[(\boldsymbol{z_{01}} - \boldsymbol{\mu_1})' \boldsymbol{\Sigma_{11}^{-1}} (\boldsymbol{z_{01}} - \boldsymbol{\mu_1})' + \frac{p}{n} \right] \frac{1}{n-p-2} + [(\boldsymbol{z_0} - \boldsymbol{\mu})' \boldsymbol{\beta} - (\boldsymbol{z_{01}} - \boldsymbol{\mu_1})' \boldsymbol{\Phi_1}]^2$$

Unconditional PMSE is

$$E[(y_0 - \tilde{y_0})^2] = \sigma_p^2 (1 + \frac{1}{n})(n-2)/(n-p-2)$$

Simulation

We established simulation to varify the conditional PMSE and unconditional PMSE for full model and subset model. We used a function $polyCor(p, rho)$ by HZhang and ZWu to generate the Covariates' variance matrix, where p is the dimension of correlation matrix and rho is the correlation between different covarites.

For Conditional PMSE

```
library(MASS)
### COnditional PMSE
ConPMSE = function(n, k, p, sigmak2, MU, SIGMA, alpha, BETA)
{
###Parameter settings
  #n : Sample size
  #k : Total number of covariates in regression
  #p : Number of covariates in subset/partial model
  #when p=k, the partial model expands into the whole model

  #sigmak2 : variance of the error term in regression
  # sigmap^2 = sigmak^2 + t(sigma)Sigma^(-1)sigma - t(sigma_1)Sigma_11^(-1)sigma_1

  #SIGMA=diag(1, k, k); #Covariates' variance matrix. Require it is positive definite.
  #SIGMA = polyCor(k, 0.3);
  #alpha : Intercept in regression
  #BETA : Coefficient vector in regression
  #MU : Covariates' mean vector
SIGMA_p = SIGMA[1:p,1:p];
MU_p=MU[1:p];
covariateNames = paste("Z", 1:k, sep=""); #Covariates' names
covariateNames_p = paste("Z", 1:p, sep="");

#true parameters calculation
sigma_00 = t(BETA)%*%SIGMA%*%BETA+sigmak2;#variance of Y
s = SIGMA%*%as.matrix(BETA);#true covariance of  Z and Y
SIGMA_11 = SIGMA[1:p,1:p];#partial variance of Z
sigmap2 = sigma_00-t(s[1:p,1])%*%solve(SIGMA_11)%*%s[1:p,1]; #variance of partial error term

simuN = 1e5; #Number of simulations
### New data

Z0 = mvrnorm(n=1, mu=MU, Sigma=SIGMA); #New covariate value
```

```
if(k>=p+1){PHI_1 = (as.matrix(BETA[1:p]))+solve(SIGMA[1:p,1:p])%*%(SIGMA[1:p,(p+1):k])%*%(as.matrix(BETA

pmse_sc = sigmak2 + sigmap2/n + sigmap2*(t(Z0[1:p]-MU[1:p])%*%solve(SIGMA[1:p,1:p])%*%(Z0[1:p]-MU[1:p])+

{pmse_fc=sigmak2*(1+1/n)+sigmak2*((Z0-MU)%*%solve(SIGMA)%*%(Z0-MU)+k/n)/(n-k-2)} #expected full model c
# loop
pmse_c = array(NA,simuN)
for (i in 1:simuN) {
  ###Estimation data
  Z = mvrnorm(n=n, mu=MU, Sigma=SIGMA);#Matrix of covariates
  EPS = rnorm(n=n, mean=0, sd=sigmak2); #Vector of errors in regression.
  Y = alpha + Z%*%BETA + EPS; #Vector of response values.
  Y0 = alpha + Z0%*%BETA + rnorm(n=1, mean=0, sd=sigmak2); #New response value.


  ###Calculations by the lm function (seems even faster than the math formula-based implementation)
  estiData = data.frame(Y, Z); #data for estimation
  names(estiData) = c("Y", covariateNames);
  estiData_p = estiData[,1:p+1]

  lmformula_p = as.formula(paste("Y ~ ", paste(covariateNames_p, collapse= "+"))); #lm model formula
  lmfit_p = lm(lmformula_p, data=estiData_p);#LSE by lm
  #Y.hat = fitted(lmfit); #Fitted reponse values


  ###Prediction
  newData_p = data.frame(t(Z0[1:p]));#New observation for prediction
  names(newData_p) = covariateNames_p;
  Y.hat.0_p = predict(lmfit_p, newData_p); #predicted response.


  ###prediction errors
  pmse_c[i] = (Y0-Y.hat.0_p)^2;
   }

if(k>=p+1){c(mean(pmse_c),pmse_sc)}
else{c(mean(pmse_c),pmse_fc)}

}
```

For Unconditional PMSE

```
####unconditional

UnconPMSE = function(n, k, p, sigmak2, MU, SIGMA, alpha, BETA)
{
SIGMA_p = SIGMA[1:p,1:p];
MU_p=MU[1:p];
covariateNames = paste("Z", 1:k, sep=""); #Covariates' names
covariateNames_p = paste("Z", 1:p, sep="");




#true parameters calculation
```

```r
sigma_00 = t(BETA)%*%SIGMA%*%BETA+sigmak2;#variance of Y
s = SIGMA%*%as.matrix(BETA);#true covariance of  Z and Y
SIGMA_11 = SIGMA[1:p,1:p];#partial variance of Z
sigmap2 = sigma_00-t(s[1:p,1])%*%solve(SIGMA_11)%*%s[1:p,1]; #variance of partial error term

simuN = 1e5; #Number of simulations

### Simulation loops
pmse = array(NA, simuN); #prediction squared errors
pme = array(NA, simuN);#prediction errors
for (i in 1:simuN) {
  ###Estimation data
  Z = mvrnorm(n=n, mu=MU, Sigma=SIGMA);#Matrix of covariates
  EPS = rnorm(n=n, mean=0, sd=sigmak2); #Vector of errors in regression.
  Y = alpha + Z%*%BETA + EPS; #Vector of response values.

  ###New data
  Z0 = mvrnorm(n=1, mu=MU, Sigma=SIGMA); #New covariate value
  Y0 = alpha + Z0%*%BETA + rnorm(n=1, mean=0, sd=sigmak2); #New response value.

  ###Calculations by the lm function (seems even faster than the math formula-based implementation)
  estiData = data.frame(Y, Z); #data for estimation
  names(estiData) = c("Y", covariateNames);
  estiData_p = estiData[,1:p+1]

  lmformula_p = as.formula(paste("Y ~ ", paste(covariateNames_p, collapse= "+"))); #lm model formula
  lmfit_p = lm(lmformula_p, data=estiData_p);#LSE by lm
  #Y.hat = fitted(lmfit); #Fitted reponse values


  ###Prediction
  newData_p = data.frame(t(Z0[1:p]));#New observation for prediction
  names(newData_p) = covariateNames_p;
  Y.hat.0_p = predict(lmfit_p, newData_p); #predicted response.


  # ###Calculations by the math formulas provided in the paper,
  # ## the results matchs with above lm-based results.
  # X = scale(Z, scale=F); #Centralized covariates
  # S <- t(X)%*%X/(n-1); #
  # s = t(X)%*%(Y-mean(Y))/(n-1);
  # sigmap[i] = sigmak^2+t(s)%*%solve(SIGMA)%*%s-t(s_p)%*%solve(SIGMA_11)%*%s_p
  # betahat = solve(S)%*%s; #Same as the coefficients given in lmfit.
  # Yhat = mean(Y) + X%*%betahat; #Same as fitted(lmfit) above.
  #
  # Zbar = apply(Z,2,mean);
  # X0 = Z0 - Zbar;
  # Y.hat.0 = mean(Y) + X0%*%betahat; #predicted response. Same as above Y.hat.0

  ###prediction errors
  pmse[i] = (Y0-Y.hat.0_p)^2;
  pme[i] = Y0-Y.hat.0_p;
```

```
}
c(mean(pmse),sigmap2*(1+1/n)*(n-2)/(n-p-2),mean(pme))
}
```

Result

From the histogram plot, we can see that both the distribution of conditional and unconditional PMSE is right-skewed and the distributions of PME are approximately standard normal. We then explored the effect of the size of the subset on the estimation accuracy. The result is shown in Table. 1.

Table 1: when n=100 and k=10

p	ConPMSE	ConEstimation	UnconPMSE	UnconEstimation
2	1.6869	1.6806	17.4111	17.4643
5	13.9333	13.9124	7.2872	7.3292
7	5.3337	5.3342	4.0888	4.1060
10	1.1212	1.1231	1.1237	1.1248

The effect of sample on estimation accuracy when we are using partial model

Table 2: when n=100 and k=5

sample size	ConPMSE	ConEstimation	UnconPMSE	UnconEstimation
50	7.2968	7.2960	7.8728	7.8408
100	5.7018	5.6880	7.3080	7.3292
200	13.0270	13.0512	7.8728	7.8408

The effect of covariates' correlation on estimation accuracy when we are using partial model

Table 3: when n=100 and k=5

correlation	ConPMSE	ConeEstimation	UnconPMSE	UnconEstimation
polyCor(10,0.3)	22.4724	22.4643	3.1196	3.1422
polyCor(10,0.5)	5.6333	5.6281	5.9457	5.9423
polyCor(10,0.8)	5.4319	5.4477	7.3442	7.3292