Laboratory 2

Ex. 1

The task is to generate orthonormal matrix of dimension 1000×950 and fit regression model:

$$Y = X\beta + \varepsilon,$$

with $\varepsilon \sim N(0, I)$ and different number of nonzero β_i.

i) Calculating the value of the tuning parameter γ for the ridge regression, so as to minimize the mean square error of the estimation of β:

$$\gamma_0 = \underset{\gamma}{arg\,min} MSE = \underset{\gamma}{arg\,min} \mathbb{E}\|\hat{\beta} - \beta\|^2$$

$$\frac{\partial MSE}{\partial \gamma} = \frac{\partial}{\partial \gamma}\left[\frac{\gamma^2}{(1+\gamma)^2}\|\beta\|^2 + \frac{p}{(1+\gamma)^2}\right] = 0 \Rightarrow \gamma_0 = \frac{p}{\|\beta\|^2}$$

ii) Calculating the bias, the varaiance and the mean squared error of this optimal estimator:

$$\hat{\beta} = \left(X^T X + \gamma I\right)^{-1} X^T Y = \frac{1}{1+\gamma} X^T Y \sim N\left(\frac{1}{1+\gamma}\beta, \frac{1}{(1+\gamma)^2} I\right)$$

$$Cov(\hat{\beta}) = \frac{1}{(1+\gamma)^2} I = \left(\frac{\|\beta\|^2}{p + \|\beta\|^2}\right)^2 I$$

$$tr(Cov(\hat{\beta})) = \frac{p}{(1+\gamma)^2}$$

$$Bias(\hat{\beta}) = \mathbb{E}\left(\hat{\beta} - \beta\right) = \frac{-\gamma}{1+\gamma}\beta = \frac{-p}{p + \|\beta\|^2}\beta$$

$$\|Bias(\hat{\beta})\|^2 = \left(\frac{-\gamma}{1+\gamma}\right)^2 \|\beta\|^2$$

$$MSE = \left(\frac{p}{p + \|\beta\|^2}\right)^2 \|\beta\|^2 + p\left(\frac{\|\beta\|^2}{p + \|\beta\|^2}\right)^2 = \frac{p\|\beta\|^2}{p + \|\beta\|^2}$$

iii) Finding the critical value and calculating power of the test based on ridge estimator:

$$H_{0i} : \beta_i = \beta_{0i} = 0, \quad H_{1i} : \beta_i \neq \beta_{0i} \neq 0$$

$$t = \left(\hat{\beta}_i - \frac{\beta_{0i}}{1+\gamma}\right)(1+\gamma) \overset{H_{0i}}{\sim} N(0,1)$$

$$c = \Phi^{-1}\left(1 - \frac{\alpha}{2}\right)$$

$$c_{FWER} = \Phi^{-1}\left(1 - \frac{\alpha}{2p}\right)$$

$$t = \left(\hat{\beta}_i - \frac{\beta_{0i}}{1+\gamma}\right)(1+\gamma) \overset{H_{1i}}{\sim} N(\beta_{1i} - \beta_{0i}, 1)$$

$$power(\beta_{1i}) = \Phi(-c - \beta_{1i}) + 1 - \Phi(c - \beta_{1i})$$

iv) Generating 200 replicates of models with different numbers of nonzero β_i and comparing bias norm, sum of variances, MSE and power between ridge and OLS methods (**??**):

	bias	tr_cov	mse	power
20	154.838	39.932	194.770	0.353
100	233.703	301.354	535.057	0.353
200	191.274	493.285	684.559	0.353

(a) Theoretical results for ridge regression.

	bias	tr_cov	mse	power	fwer
20	5.101	953.777	954.109	0.360	0.085
100	4.879	948.361	948.498	0.354	0.090
200	4.203	951.731	951.176	0.352	0.075

(b) Results for OLS.

	bias	tr_cov	mse	power	fwer
20	147.315	47.915	194.991	0.480	0.345
100	144.530	410.945	553.419	0.572	0.590
200	77.350	651.341	725.435	0.544	0.440

(c) Results for ridge (glmnet).

	bias	tr_cov	mse	power	fwer
20	154.636	40.091	194.526	0.360	0.085
100	234.373	300.834	533.703	0.354	0.090
200	195.250	494.184	686.963	0.352	0.075

(d) Results for ridge (by hand).

Table 1.1: Theoretical and empirical norm of bias, trace of covariance, MSE and power for different methods.

The table 1.1 presents results of those replications. Theoretical results are almost identical with ridge regression calculated "by hand". Using library glmnet gives worse results, among them not holding FWER on 0.1 level. Power of OLS is equal to ridge, because of modification in testing (OLS is equivalent to ridge regression with $\gamma = 0$) so that it also holds FWER on 0.1 level. Testing OLS estimators using γ for corresponding ridge regression, gives power close to 1. The biggest difference between OLS and ridge is the MSE. It's much smaller for ridge, especialy when there is not a lot of nonzero elements. MSE of OLS stays constant and its estimators have larger variances, but have lower biases.

Ex. 2

The task is to generate matrix $X_{1000\times 950}$ such that its elements are iid random variables from $N(0, 1/\sqrt{n})$ and then generating response variables like in previous task, using ridge regression with parameter choosen by minimizing SURE criterion and cross-validation, and OLS regression with model selected by MBIC2. Repeating experiment 100 times and comparing mean square errors of estimation of β and $\mu = X\hat{\beta}$:

k	$MSE(\beta)$	$MSE(\mu)$
20	202.727	173.743
100	672.375	415.725
200	1074.386	526.216

(a) Results for SURE.

k	$MSE(\beta)$	$MSE(\mu)$
20	245.000	247.359
100	1224.970	1267.415
200	2449.068	2447.469

(b) Results for CV.

k	$MSE(\beta)$	$MSE(\mu)$
20	20022.413	955.572
100	19927.649	954.405
200	19835.352	942.429

(c) Results for OLS.

k	$MSE(\beta)$	$MSE(\mu)$
20	187.255	178.614
100	1111.707	940.615
200	2468.711	2368.997

(d) Results for MBIC2.

Table 2.1: Results of MSE for different regression models with $\beta_1 = \ldots = \beta_k = 3.5$.

The table 2.1 presents results of those replications. We can infere, that MSE is always the lowest for ridge regression that minimizes prediction error (PE), but it's also the slowest one, because it needs a lot of computation to find right parameter γ. Also, the OLS method is always the worst one, but its always constant in its errors. Calculating ridge regression using cross-validation is quicker, but slightly worse than SURE mehtod. Using OLS with mbic2 criterion is close to CV method.

Ex. 3

The task is to repeat the previous task, but with $\beta_1 = \ldots = \beta_k = 5$. Repeating experiment 100 times and comparing mean square errors of estimation of β and $\mu = X\hat{\beta}$:

k	$MSE(\beta)$	$MSE(\mu)$
20	352.566	273.389
100	1150.254	520.090
200	1691.578	624.688

(a) Results for SURE.

k	$MSE(\beta)$	$MSE(\mu)$
20	500.000	539.647
100	2498.567	2580.582
200	4998.986	5277.578

(b) Results for CV.

k	$MSE(\beta)$	$MSE(\mu)$
20	18474.741	950.603
100	18093.470	948.889
200	19310.568	949.193

(c) Results for OLS.

k	$MSE(\beta)$	$MSE(\mu)$
20	66.609	63.256
100	322.703	249.143
200	4592.811	3050.850

(d) Results for MBIC2.

Table 3.1: Results of MSE for different regression models with $\beta_1 = \ldots = \beta_k = 5$.

The table 3.1 presents results of those replications. The OLS method has not changed since the previous task. Every other MSE is bigger than in previous task, except the case of $k = 100$ for MBIC2. This could be, because of using function *stepwise* instead of *fast_forward*, which is used later for shortening the time of computations. Function *stepwise* is more accurate and gives better results.

Ex. 4

The task is to repeat task 2 and 3 when rows of X are iid random vectors from $\frac{1}{n}N(0, \Sigma)$, where $\Sigma_{ii} = 1$ and for $i \neq j$, $\Sigma_{ij} = 0.5$. Repeating experiment 100 times and comparing mean square errors of estimation of β and $\mu = X\hat{\beta}$:

a) $\beta_1 = \ldots = \beta_k = 3.5$

k	$MSE(\beta)$	$MSE(\mu)$
20	258.676	1.474
100	1120.830	2.975
200	1915.784	6.862

(a) Results for SURE

k	$MSE(\beta)$	$MSE(\mu)$
20	245.000	2.568
100	1225.000	61.099
200	2439.924	238.877

(b) Results for CV

k	$MSE(\beta)$	$MSE(\mu)$
20	36630614.619	942.881
100	37688017.134	948.485
200	37694027.202	953.357

(c) Results for OLS

k	$MSE(\beta)$	$MSE(\mu)$
20	2766.777	3.848
100	42878.220	22.934
200	85258.867	43.093

(d) Results for MBIC2

Table 4.1: Results of MSE for different regression models with $\beta_1 = \ldots = \beta_k = 3.5$ and $X_{ij} \sim \frac{1}{n}N(0, \Sigma)$.

k	$MSE(\beta)$	$MSE(\mu)$
20	271.406	136.201
100	752.439	335.354
200	5625.719	817.840

(a) Results for SURE

k	$MSE(\beta)$	$MSE(\mu)$
20	239.243	372.386
100	1100.399	7239.309
200	1961.677	27212.738

(b) Results for CV

k	$MSE(\beta)$	$MSE(\mu)$
20	38868.759	947.290
100	38036.446	945.243
200	38314.439	951.272

(c) Results for OLS

k	$MSE(\beta)$	$MSE(\mu)$
20	139.928	66.171
100	682.457	300.042
200	5049.350	1974.668

(d) Results for MBIC2

Table 4.2: Results of MSE for different regression models with $\beta_1 = \ldots = \beta_k = 3.5$ and $X_{ij} \sim \frac{1}{\sqrt{n}}N(0, \Sigma)$.

b) $\beta_1 = \ldots = \beta_k = 5$

k	$MSE(\beta)$	$MSE(\mu)$
20	513.169	1.383
100	2235.171	4.339
200	3854.293	11.808

(a) Results for SURE

k	$MSE(\beta)$	$MSE(\mu)$
20	500.000	5.276
100	2499.264	127.265
200	4943.586	479.936

(b) Results for CV

k	$MSE(\beta)$	$MSE(\mu)$
20	35474733.111	946.731
100	35860809.621	949.790
200	35080103.264	952.296

(c) Results for OLS

k	$MSE(\beta)$	$MSE(\mu)$
20	5783.030	6.712
100	61965.378	31.666
200	120910.311	59.190

(d) Results for MBIC2

Table 4.3: Results of MSE for different regression models with $\beta_1 = \ldots = \beta_k = 5$ and $X_{ij} \sim \frac{1}{n}N(0, \Sigma)$.

k	$MSE(\beta)$	$MSE(\mu)$
20	482.524	208.019
100	1757.673	487.038
200	12147.574	1035.190

(a) Results for SURE

k	$MSE(\beta)$	$MSE(\mu)$
20	488.210	765.801
100	2245.697	14803.057
200	4009.769	55451.363

(b) Results for CV

k	$MSE(\beta)$	$MSE(\mu)$
20	38538.776	947.913
100	38274.305	944.235
200	38533.282	949.673

(c) Results for OLS

k	$MSE(\beta)$	$MSE(\mu)$
20	107.194	52.515
100	1255.631	522.065
200	9876.675	4440.676

(d) Results for MBIC2

Table 4.4: Results of MSE for different regression models with $\beta_1 = \ldots = \beta_k = 5$ and $X_{ij} \sim \frac{1}{\sqrt{n}}N(0, \Sigma)$.

The tables 4.1 and 4.3 present results of those replications. Each experiment was also repeated twice for $X_{ij} \sim \frac{1}{n}N(0, \Sigma)$ and $X_{ij} \sim \frac{1}{\sqrt{n}}N(0, \Sigma)$ to check if the task contained error. Usually we divide by $\sqrt{n}$ to make norm L_2 of each column around 1. Seems that this difference have mostly influence on $MSE(\mu)$, which is bigger when diving by square root.

Results throughout the experiments show that in those cases OLS mehtod is always the worst one and SURE method is the best, but also the slowest when comparing $MSE(\beta)$. OLS always has the same $MSE(\mu)$. CV mehtod is worse with growing number of nonzero elements than SURE. In some cases MBIC2 method could be better than CV (lower k).

Appendices

Ex. 1

Listing 1.1: Generating data.

```
generate_design <- function(n=1000, p=950) {
  X = matrix(rnorm(n*p, 0, 1/sqrt(n)), n, p)
  return(X)
}

generate_response <- function(X, beta=3.5, nonzero=20, rdist=rnorm) {
  betas = c(rep(beta,nonzero), rep(0,ncol(X)-nonzero))
  X%*%betas + rdist(nrow(X))
}
```

Listing 1.2: Theoretical and empirical statistics.

```
#i)
norm2 <- function(X) sum(X^2)
gamma_min <- function(beta) length(beta)/norm2(beta)

#ii)
bias_thr <- function(beta, gamma) -gamma/(1+gamma) * beta
bias_norm_thr <- function(beta, gamma) (gamma/(1+gamma))^2 * norm2(beta)
bias_norm_est <- function(beta, beta_hat) norm2(beta_hat - beta)

cov_beta_thr <- function(beta, gamma) 1/(1+gamma)^2 * diag(length(beta))
cov_trace_thr <- function(beta, gamma) length(beta)/(1+gamma)^2
cov_trace_est <- function(beta_hat) sum(diag(cov(beta_hat,beta_hat)))

mse_thr <- function(beta, gamma) bias_norm_thr(beta, gamma) + cov_trace_thr
    (beta, gamma)
se <- function(beta, beta_hat) sum((beta_hat - beta)^2)

#iii)
power_thr <- function(beta1, p=950, alpha=.1) {
  c = qnorm(1-alpha/(2*p))
  pnorm(-c - beta1) + 1 - pnorm(c - beta1)
}
power_est <- function(beta, beta_hat, gamma, p=950, alpha=.1) {
  c = qnorm(1-alpha/(2*p)) / (1+gamma)
  test = abs(beta_hat[beta!=0]) > c
  sum(test) / sum(beta!=0)
}

fwer_est <- function(beta, beta_hat, gamma, p=950, alpha=.1) {
  c = qnorm(1-alpha/(2*p)) / (1+gamma)
  test = abs(beta_hat[beta==0]) > c
  sum(test)>0
}
```

Listing 1.3: Empiric comparision of models

```
#iv)
ex1 <- function(n=1000, p=950, beta=3.5, nonzero=c(20,100,200), rep=200) {
  beta_hat=matrix(0,rep,p); OLS=matrix(0,rep,p); ridge=matrix(0,rep,p);
  beta_hat_results=data.frame(); OLS_results=data.frame(); ridge_results=
      data.frame();
```

```
  results_list = list(beta_hat_results, OLS_results, ridge_results)
  names(results_list) = c("OLS", "ridge", "beta_hat")

  X = randortho(n)[,1:p]#X = generate_design(n,p,beta,max(nonzero))
  for (k in nonzero) {
    betas = c(rep(beta,k), rep(0,p-k))
    gamma = gamma_min(betas)
    for (r in 1:rep) {
      Y = X%*%betas + rnorm(n)
      OLS[r,] = coef(lm(Y~X-1))
      ridge[r,] = coef(glmnet(X, Y, lambda=gamma/n, alpha = 0, intercept=
          FALSE, standardize = FALSE))[-1,1]
      beta_hat[r,] = 1/(1+gamma) * t(X) %*% Y
      cat("k=␣",k,"r=␣",r,"\n")
    }

    est_list = list(OLS, ridge, beta_hat)
    for (i in 1:length(est_list)) {
      est = est_list[[i]]
      bias = bias_norm_est(betas, apply(est, MARGIN=2, mean))
      tr_cov = sum(apply(est, MARGIN=2, var))
      mse = mean(apply(est, MARGIN=1, function(b)se(betas, b)))

      power = mean(apply(est, MARGIN=1, function(b)power_est(betas,b,
          ifelse(i==1, 0, gamma))))
      fwer = mean(apply(est, MARGIN=1, function(b)fwer_est(betas,b, ifelse(
          i==1, 0, gamma))))

      results_list[[i]] = rbind(results_list[[i]], c(bias, tr_cov, mse,
          power, fwer))
    }

  }

  save(list=c("results_list"), file=paste0("SL_lab2_ex1",".RData"))
  for (i in 1:length(results_list)) {
    name = names(results_list)[i]
    colnames(results_list[[i]]) = c("bias", "tr_cov", "mse", "power", "fwer
        ")
    rownames(results_list[[i]]) = paste(nonzero)
    printTable(results_list[[i]], paste("Results␣for", name), paste0("SL_
        lab2_ex1_",name))
  }

  return(results_list)
}
```

Listing 1.4: Theoretical values of ridge regression.

```
ex1_thr <- function(n=1000, p=950, beta=3.5, nonzero=c(20,100,200)) {
  results_thr = data.frame()
  for (k in nonzero) {
    betas = c(rep(beta,k), rep(0,p-k))
    gamma = gamma_min(betas)
    bias = bias_norm_thr(betas,gamma)
    tr_cov = cov_trace_thr(betas,gamma)
    mse = mse_thr(betas,gamma)
    power = power_thr(beta, p)
    results_thr = rbind(results_thr, c(bias, tr_cov, mse, power))
```

```
  }

  colnames(results_thr) = c("bias", "tr_cov", "mse", "power")
  rownames(results_thr) = paste(nonzero)
  save(list=c("results_thr"), file=paste("SL_lab2_ex1_thr",".RData", sep=""
      ))
  printTable(results_thr, "Theoretical results", "SL_lab2_ex1_thr")
  return(results_thr)
}
```

Ex. 2

Listing 2.1: Additional functions.

```
SE <- function(Yhat, Y) norm2(Y - Yhat)

PE_est <- function(RSS, eigen, gamma, sigma=1) {
  RSS + 2*sigma^2 * sum( eigen / (eigen + gamma))
}
```

Listing 2.2: Ridge coefficients minimizinig PE.

```
ridge_SURE_min2 <- function(X,Y) {
  beta = c(rep(3.5,20), rep(0,930))
  lambdaseq = seq(from=0.0001,to=0.005, length.out = 200)
  values = eigen(t(X)%*%X, only.values = TRUE)$values
  n = nrow(X)
  obj = glmnet(X,Y,alpha=0,intercept=FALSE, standardize=FALSE, lambda=
      lambdaseq)
  betaridge = obj$beta
  lambdas = obj$lambda

  sureridge = NULL
  for (u in 1:ncol(betaridge))
    sureridge[u] = PE_est(sum((X%*%betaridge[,u]-Y)^2), values, lambdas[u]*
        n)

  ind = which.min(sureridge)
  return(betaridge[,ind])
}
```

Listing 2.3: OLS coefficients choosen by mbic2.

```
ols_mbic2 <- function(X,Y) {
  model = as.numeric(stepwise(prepare_data(Y,X),crit=mbic2)$model)
  beta =  rep(0, ncol(X))
  if(length(model)!=0)
    beta[model] = coef(lm(Y~X[,model]-1))
  return(beta)
}
```

Listing 2.4: Empiric comparision of models.

```
ex2 <- function(n=1000, p=950, beta=3.5, nonzero=c(20,100,200), rep=100,
    generate_x=generate_design) {
  SURE = matrix(0, rep, p); CV = matrix(0, rep, p); OLS = matrix(0, rep, p)
      ; MBIC2 = matrix(0, rep, p);
  results_list = list(data.frame(), data.frame(), data.frame(), data.frame
      ())
  names(results_list) = c("SURE", "CV", "OLS", "MBIC2")
```

```
  X = generate_x(n,p)
  for (k in nonzero) {
    betas = c(rep(beta,k), rep(0,p-k))
    for (r in 1:rep) {
      Y = X%*%betas + rnorm(n)
      SURE[r,] = ridge_SURE_min2(X,Y)
      CV[r,] = coef(cv.glmnet(X,Y,alpha=0,intercept=FALSE, standardize=
          FALSE))[-1,1]
      OLS[r,] = coef(lm(Y~X-1))
      MBIC2[r,] = ols_mbic2(X,Y)
      cat("k= ",k,"r= ",r,"\n")
    }

    est_list = list(SURE, CV, OLS, MBIC2)
    for (i in 1:length(est_list)) {
      est = est_list[[i]]
      mse = mean(apply(est, 1, function(b)norm2(b-betas)))
      mu = mean(apply(est, 1, function(b)norm2(X%*%(b-betas))))
      results_list[[i]] = rbind(results_list[[i]], c(mse,mu))
    }

  }

  save(list=c("results_list"), file=paste0("SL_lab2_ex3",".RData"))
  for (i in 1:length(results_list)) {
    name = names(results_list)[i]
    colnames(results_list[[i]]) = c("mse(beta)", "mse(mu)")
    rownames(results_list[[i]]) = paste(nonzero)
    printTable(results_list[[i]], paste("Results for", name), paste0("SL_
        lab2_ex3_",name))
  }

  return(results_list)
}
```

Ex. 3

Listing 3.1: Empiric comparision of models.

```
ex2(beta = 5) #REMBEMER FILE NAME
```

Ex. 4

Listing 4.1: Generating data.

```
generate_mvrnorm <- function(n=1000, p=950, ro=0.5) {
  Sigma = matrix(ro, p, p)
  diag(Sigma) = 1
  X = mvrnorm(n, rep(0,p), Sigma)
}
```

Listing 4.2: Empiric comparision of models.

```
ex2(generate_x = generate_mvrnorm) #REMEMBER FILE NAME
ex2(beta = 5, generate_x = generate_mvrnorm)
```