Assignment 2 Solution

Can Yang

Department of Mathematics, The Hong Kong University of Science and Technology

Problem 1

$$\hat{\beta} = \arg\min\{\|b\|_2 : b \text{ minimizes } \frac{1}{2n}\|y - Xb\|_2^2\}. \tag{1}$$

(a) When $n \geq p$, problem 1 has unique solution $\hat{\beta} = (X^T X)^{-1} X^T y$ because X has full rank. When $n < p$, $\hat{\beta} = X^T (XX^T)^{-1} y$ clearly minimizes $\frac{1}{2n}\|y - Xb\|_2^2$. We only need to show $\hat{\beta} = X^T (XX^T)^{-1} y$ has the minimum ℓ_2 norm among all solutions. Let $\tilde{\beta}$ be a solution of $X\tilde{\beta} = y$. Clearly, $X(\tilde{\beta} - \hat{\beta}) = 0$ and

$$\begin{aligned}(\tilde{\beta} - \hat{\beta})^T \hat{\beta} &= (\tilde{\beta} - \hat{\beta})^T X^T (XX^T)^{-1} y \\ &= (X(\tilde{\beta} - \hat{\beta}))^T (XX^T)^{-1} y \\ &= 0,\end{aligned}$$

i.e., $(\tilde{\beta} - \hat{\beta}) \perp \hat{\beta}$. So we have

$$\|\tilde{\beta}\|^2 = \|\hat{\beta} + \tilde{\beta} - \hat{\beta}\|^2 = \|\hat{\beta}\|^2 + \|\tilde{\beta} - \hat{\beta}\|^2 \geq \|\hat{\beta}\|^2,$$

i.e., β has the minimum ℓ_2 norm among all solutions.

The degrees of freedom

$$df = \mathbb{E}\left[\sum_i \frac{\partial \hat{y}_i}{\partial y_i}\right] = trace(X^T (X^T X)^{-1} X) = trace(XX^T (XX^T)^{-1}) = trace(I_p) = p.$$

(b)

We consider $\|\beta\|^2 = r$.

```
min2norm <- function(X,y){
  n <- dim(X)[1]
  p <- dim(X)[2]
  if(p <= n){
    XX <- t(X)%*%X
    XX <- (XX + t(XX)) / 2
    betahat <- solve(XX) %*% (t(X) %*% y)
  }else{
    XX <- X%*%t(X)
    XX <- (XX + t(XX)) / 2
    betahat <- t(X) %*% (solve(XX) %*% y)
  }
  return(betahat)
}



n <- 200
ntest <- 1000
nrep <- 50
gamma <- 4
SNR <- 1
Q = seq(11,gamma*n,5)

err_train <- matrix(0, nrow = nrep, ncol = length(Q))
err_test <- matrix(0, nrow = nrep, ncol = length(Q))
h2 <- matrix(0, nrow = nrep, ncol = length(Q))

ytesthat <- matrix(0, nrow = ntest, ncol = nrep)
biasSQ <- matrix(0,nrow = 1, ncol = length(Q))
variance <- matrix(0,nrow = 1, ncol = length(Q))

#p=10

ind = 0
for(p in Q){
  ind = ind + 1
  beta <- matrix(rnorm(p),p,1)
  beta <- beta/sqrt(sum(beta^2))*sqrt(SNR)

  X <- matrix(rnorm(n*p), nrow = n, ncol = p)
  Xtest <- matrix(rnorm(ntest*p), nrow = ntest, ncol = p)
  y0<- X%*%beta
  ytest0 <- Xtest%*%beta

  for(irep in 1:nrep){
    y <- y0 + rnorm(n)
    h2[irep, ind] = (var(y0)/var(y))
    ytest0 = Xtest%*%beta
    ytest <- ytest0 + rnorm(ntest)

    betahat <- min2norm(X,y)
    yhat = X%*%betahat
```

```
    ytesthat[,irep] = Xtest%*% betahat
    err_train[irep, ind] <- sum((yhat - y)^2) / (n)
    err_test[irep, ind] <- sum((ytesthat[,irep] - ytest)^2) / (ntest)
  }
  # compute mean of predicted values
  y_bar <- rowMeans(ytesthat)                 # E(f^hat)
  # compute bias^2
  biasSQ[ind] <- mean((ytest0-y_bar)^2)       # E[ (f - E(f^hat))^2 ]
  # compute variance
  variance[ind] <- mean((ytesthat-y_bar)^2)   # E[ (E(f^hat) - f^hat)^2 ]

  print(p)
}

filepath <- "/Users/canyang/Documents/current/D-Drive/Teaching/
            ComputerAgeStatisticalInference/Assignment/"
save(err_test,biasSQ,variance,file=paste0(filepath,"out_n",n,
                                          "_gamma",gamma,"_SNR",SNR,".RData"))



library(ggplot2)
n <- 200
gamma <- 4
SNR <- c(1,2,3,5,7)
Q = seq(11,gamma*n,5)

out_err <- out_biasSQ <- out_variance <- data.frame()
filepath<-"/Users/canyang/Documents/current/D-Drive/Teaching/ComputerAgeStatisticalInference/Assignment/
for(i in 1:length(SNR)){
  load(paste0(filepath,"out_n",n,"_gamma",gamma,"_SNR",SNR[i],".RData"))
  out_err <- rbind(out_err,cbind(risk=colMeans(err_test),gamma=Q/n,SNR=SNR[i]))
  out_biasSQ <- rbind(out_biasSQ,cbind(biasSQ=c(biasSQ),gamma=Q/n,SNR=SNR[i]))
  out_variance <- rbind(out_variance,cbind(variance=c(variance),gamma=Q/n,SNR=SNR[i]))
}
out_err$SNR <- as.factor(out_err$SNR)
out_biasSQ$SNR <- as.factor(out_biasSQ$SNR)
out_variance$SNR <- as.factor(out_variance$SNR)

P_err <- ggplot(out_err,aes(x=gamma,y=risk,color=SNR))+geom_point()+geom_line()+
  coord_cartesian(ylim=c(0,10))+theme_bw()+ theme(legend.position="top")
P_err
ggsave(P_err,file=paste0(filepath,"/risk.pdf"),width = 5,height=4)

P_bias <- ggplot(out_biasSQ,aes(x=gamma,y=biasSQ,color=SNR))+geom_point()+geom_line()+
  coord_cartesian(ylim=c(0,7))+theme_bw()+ theme(legend.position="top")
P_bias
ggsave(P_bias,file=paste0(filepath,"/bias.pdf"),width = 5,height=4)

P_variance <- ggplot(out_variance,aes(x=gamma,y=variance,color=SNR))+geom_point()+geom_line()+
  coord_cartesian(ylim=c(0,3))+theme_bw()+ theme(legend.position="top")
P_variance
ggsave(P_variance,file=paste0(filepath,"/variance.pdf"),width = 5,height=4)
```

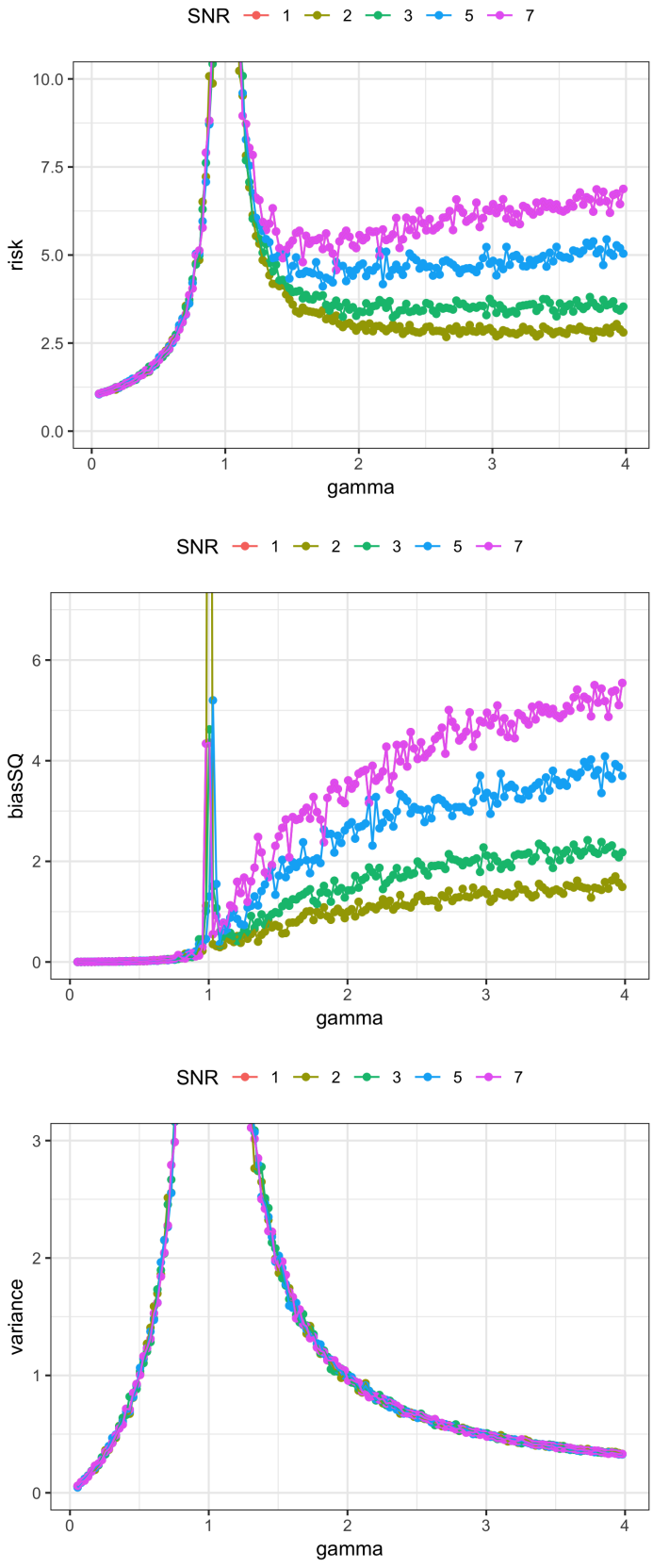

Figure 1: From top to bottom: Risk $\|y_{test} - X_{test}\hat{\beta}\|^2$, Squared bias $[X\beta - \mathbb{E}_D(X\hat{\beta})]^2$ and Variance $\mathbb{E}\left[[\mathbb{E}_D(X\hat{\beta}) - X\hat{\beta}]^2\right]$.

(c) When $n \geq p$, it is obvious that the gradient decent algorithm converges to $\hat{\beta} = (X^T X)^{-1} X^T y$. Here we consider the case $n < p$. Let the singular value decomposition (SVD) of X be

$$X = U\Sigma V^T = U \begin{bmatrix} \Sigma_1 & 0 \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix} = U\Sigma_1 V_1^T$$

The min-norm solution can be rewritten as

$$\hat{\beta} = X^T (XX^T)^{-1} y = \cdots = V_1 \Sigma_1^{-1} U^T y.$$

The gradient of $\frac{1}{2}\|y - Xb\|^2$ is given as $g(b) = -X^T(y - Xb)$. By gradient descent with stepsize $\epsilon > 0$,

$$\begin{aligned} \beta_{k+1} &= \beta_k - \epsilon \cdot g(b)\Big|_{b=\beta_k} = \beta_k + \epsilon \cdot X^T (y - X\beta_k) \\ &= (I - \epsilon X^T X)\beta_k + \epsilon \cdot X^T y. \end{aligned}$$

Hence, we have

$$\beta_k = (I - \epsilon X^T X)^k \beta_0 + \epsilon \cdot \sum_{j=0}^{k-1} (I - \epsilon \cdot X^T X)^j X^T y.$$

Denoting $\tilde{\beta}_k = V^T \beta_k$, we can rewrite

$$\begin{aligned} \tilde{\beta}_k &= (I - \epsilon \cdot \Sigma^T \Sigma)^k \tilde{\beta}_0 + \epsilon \cdot \sum_{j=0}^{k-1} (I - \Sigma^T \Sigma)^j \Sigma^T U^T y \\ &= \begin{bmatrix} (I - \epsilon \Sigma_1^2)^k & 0 \\ 0 & I \end{bmatrix} \tilde{\beta}_0 + \epsilon \sum_{j=0}^{k-1} \begin{bmatrix} (I - \epsilon \Sigma_1^2)^j & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} \Sigma_1 \\ 0 \end{bmatrix} U^T y \\ &= \begin{bmatrix} (I - \epsilon \Sigma_1^2)^k & 0 \\ 0 & I \end{bmatrix} \tilde{\beta}_0 + \epsilon \sum_{j=0}^{k-1} \begin{bmatrix} (I - \epsilon \Sigma_1^2)^j \Sigma_1 \\ 0 \end{bmatrix} U^T y \end{aligned}$$

Choosing $0 < \epsilon < 1/\lambda_{\max}(X^T X)$ such that all eigenvalues are strictly inside the unit circle, we have

$$\tilde{\beta}_k \to \tilde{\beta}_\infty = \begin{bmatrix} 0 & 0 \\ 0 & I \end{bmatrix} \tilde{\beta}_k + \epsilon \sum_{j=0}^{\infty} \begin{bmatrix} (I - \epsilon \Sigma_1^2)^j \Sigma_1 \\ 0 \end{bmatrix} U^T y$$

where

$$\epsilon \sum_{j=0}^{\infty} (I - \epsilon \Sigma_1^2)^j \Sigma_1 = \epsilon (I - I + \epsilon \Sigma_1^2)^{-1} \Sigma_1 = \Sigma_1^{-1}$$

Therefore,

$$\tilde{\beta}_\infty = \begin{bmatrix} 0 & 0 \\ 0 & I \end{bmatrix} \tilde{\beta}_0 + \begin{bmatrix} \Sigma_1^{-1} \\ 0 \end{bmatrix} U^T y.$$

Recall that $\beta = V\tilde{\beta}$,

$$\beta_\infty = V_2 V_2^T \beta_0 + \underbrace{V_1 \Sigma_1^{-1} U^T y}_{\hat{\beta}}.$$

In conclusion, if β_0 is the vector of zero or orthogonal to the null space of X, then gradient descent will converge to the minimum ℓ_2 norm solution.

(d) The exact solution to the gradient flow is

$$\hat{\beta}(t) = (X^T X)^{-1}(I - \text{expm}(-tX^T X/n))X^T y, \tag{2}$$

for all $t \geq 0$, where $\text{expm}(A) = I + A + A^2/2 + A^3/3! + \ldots$ is the matrix exponential. This solution can be verifited by differentiating (2) and using basic properties of matrix exponential.

The matrix exponential $\text{expm}(A)$ can be exactly computed as $\text{expm}(A) = U\text{diag}(\exp(D))U^T$, where A is a $p \times p$ matrix, U is the matrix collecting all the $p$eigenvectors and $D = [d_1, \ldots, d_p] \in \mathbb{R}^p$ collects all the eigenvalues.

Let $X^T X/n = U\text{diag}(D)U^T$. Then we have

$$\begin{aligned} I - \exp(-tX^T X/n) &= I - U\text{diag}(\exp(-tD))U^T = U(I - \text{diag}(\exp(-tD)))U^T, \\ (X^T X)^{-1} &= \frac{1}{n} U\text{diag}(D^{-1})U^T, \end{aligned}$$

where the inverse and exp operators are taken in an elementwise manner. Therefore, (2) can be written as

$$\begin{aligned} \hat{\beta}(t) &= \frac{1}{n} U\text{diag}(D^{-1})(I - \text{diag}(\exp(-tD)))U^T X^T y \\ &= \frac{1}{n} U\text{diag}(D^{-1} - D^{-1}\exp(-tD))U^T X^T y. \end{aligned}$$

(e) The solution paths of Ridge regression and Gradient flow are often very similar to each other. This suggests that there exists imexplicit regularization when using gradient decent or gradient flow.

```
set.seed(1)

lm_ridge <- function(X,y,lam){
  p <- ncol(X)

  tmp <- solve(t(X)%*%X+lam*diag(p)) %*% t(X)
  Hat <- X %*% tmp
  yhat <- Hat %*% y
  beta <- tmp%*%y
```

```
  ret <- list(yhat=yhat,Hat=Hat,beta=beta)
  return(ret)
}

lm_gf <- function(X,y,t){
  p <- ncol(X)
  eigenXX <- eigen(t(X)%*%X)
  D <- eigenXX$values
  U <- eigenXX$vectors
  D1 <- 1/D-exp(-t*D)/D

  beta <- U%*%(D1*(t(U)%*%(t(X)%*%y)))
  return(list(beta=beta))
}



n <- 50
ntest <- 300
p <- 10

nrep <- 30
nlam <- 50   # number of lambdas for ridge
nt <- 100   # number of lambdas for ridge

sb <- 0.5
s2 <- 1-sb

beta <- rnorm(p,0,sqrt(sb))

X <- matrix(rnorm(n*p),n,p)/sqrt(p)
y0 <- X%*%beta

D <- eigen(t(X)%*%X)$values
loglam <- seq(3*log10(max(D)),-2*log10(max(D)),length.out = nlam)
lam_seq <- 10^(loglam)

logt <- seq(-3*log10(max(D)),2*log10(max(D)),length.out = nt)
t_seq <- 10^(logt)#seq(0,10,length.out = nt)

Xtest <- matrix(rnorm(ntest*p),ntest,p)/sqrt(p)
ytest0 <- Xtest%*%beta


beta_ridge <- array(0,dim=c(nrep,p,nlam))
beta_gf <- array(0,dim=c(nrep,p,nt))

yhat_ridge <- array(0,dim = c(nrep,ntest,nlam))
yhat_gf <- array(0,dim = c(nrep,ntest,nt))

err_ridge <- matrix(0,nrep,nlam)
err_gf <- matrix(0,nrep,nt)
```

```
for(i in 1:nrep){
  e <- rnorm(n,0,sqrt(s2))
  y <- y0+e
  data_mat <- data.frame(y=y,X=X)

  ytest <- ytest0 + rnorm(ntest,0,sqrt(s2))

  # fit ridge
  for(j in 1:nlam){
    fit_ridge <- lm_ridge(X,y,lam_seq[j])
    yhat_ridge[i,,j] <- Xtest%*%fit_ridge$beta
    beta_ridge[i,,j] <- fit_ridge$beta
  }
  err_ridge[i,] <- colMeans((yhat_ridge[i,,]-c(ytest))^2)


  # fit gradient flow
  for(j in 1:nt){
    fit_gf <- lm_gf(X,y,t_seq[j])
    yhat_gf[i,,j] <- Xtest%*%fit_gf$beta
    beta_gf[i,,j] <- fit_gf$beta
  }
  err_gf[i,] <- colMeans((yhat_gf[i,,]-c(ytest))^2)

  cat(i,"-th trial finished.\n",sep="")

}

par(mfrow=c(2,2))

matplot(y=t(beta_ridge[1,,]),
        x=colSums(beta_ridge[1,,]^2)/max(colSums(beta_ridge[1,,]^2)),
        type = "l",ylab="beta",xlab="||beta||/max(||beta||)",
        main="Solution path of Ridge")
matplot(y=t(beta_gf[1,,]),
        x=colSums(beta_gf[1,,]^2)/max(colSums(beta_gf[1,,]^2)),
        type = "l",ylab="beta",xlab="||beta||/max(||beta||)",
        main="Solution path of Gradient Flow")

# compute mean of predicted values
ybar_ridge <- apply(yhat_ridge,c(2,3),mean)                # E(f^hat)
# compute bias^2
biasSQ_ridge <- colMeans((ybar_ridge-c(ytest0))^2)       # E[ (f - E(f^hat))^2 ]
# compute variance
variance_ridge <- apply((yhat_ridge-aperm(replicate(nrep,ybar_ridge),
                                          c(3,1,2)))^2,3,mean)   # E[ (E(f^hat) - f^hat)^2 ]

# compute mean of predicted values
ybar_gf <- apply(yhat_gf,c(2,3),mean)                 # E(f^hat)
# compute bias^2
biasSQ_gf <- colMeans((ybar_gf-c(ytest0))^2)       # E[ (f - E(f^hat))^2 ]
# compute variance
variance_gf <- apply((yhat_gf-aperm(replicate(nrep,ybar_gf),
```

```
                                    c(3,1,2)))^2,3,mean)   # E[ (E(f^hat) - f^hat)^2 ]

ylim <- range(c(biasSQ_ridge,variance_ridge))
plot(-loglam,biasSQ_ridge,ylim=ylim,type="l",col="blue",
     main="Ridge regression",ylab="error",xlab="-log(lambda)")
lines(-loglam,variance_ridge,col="red")
lines(-loglam,biasSQ_ridge+variance_ridge,col="green")

legend("topright", legend=c("biasSQ","variance","total error"),
       col=c("blue", "red","green"), lty=1, cex=0.8)

ylim <- range(c(biasSQ_gf,variance_gf))
plot(logt,biasSQ_gf,ylim=ylim,type="l",col="blue",main="Gradient Flow",
     ylab="error",xlab="log(t)")
lines(logt,variance_gf,col="red")
lines(logt,biasSQ_gf+variance_gf,col="green")

legend("topright", legend=c("biasSQ","variance","total error"),
       col=c("blue", "red","green"), lty=1, cex=0.8)
```

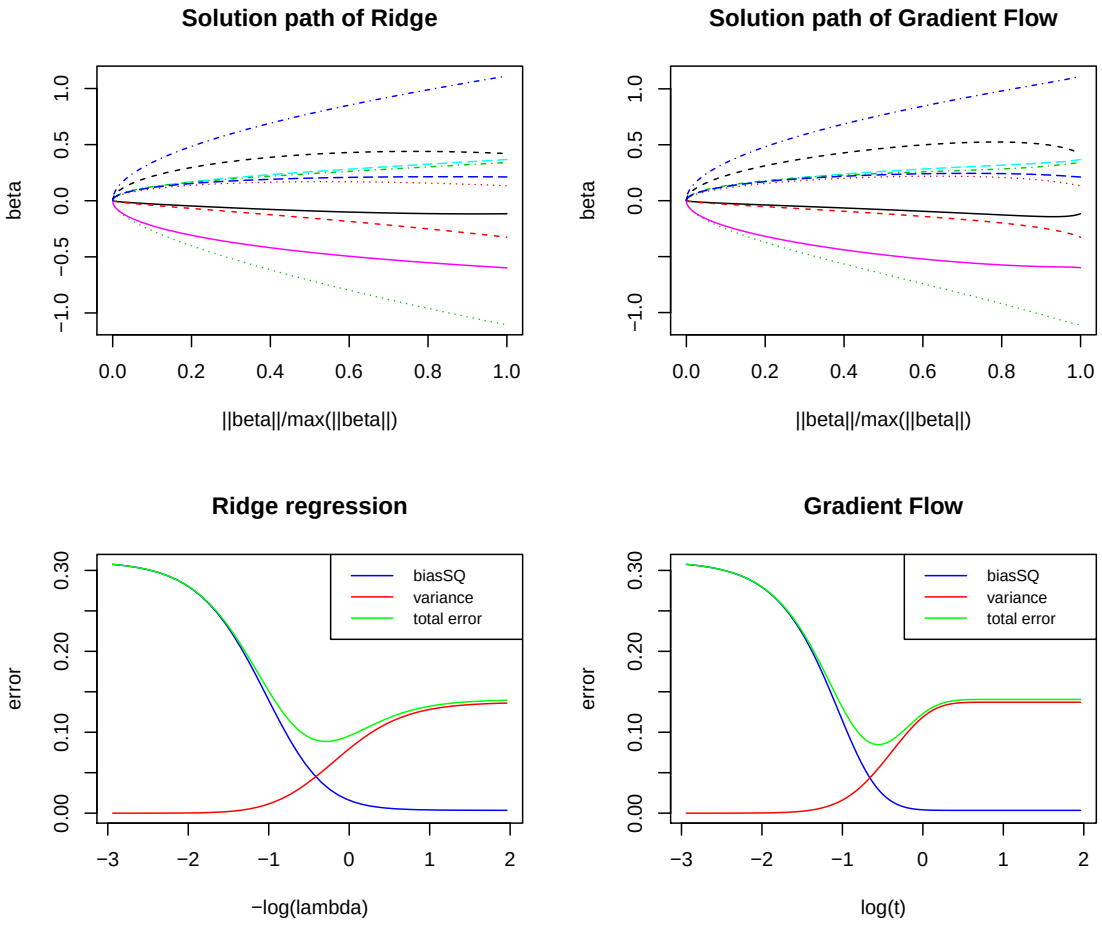

Figure 2: Ridge regression vs. Gradient flow

Problem 2.

(a) The probabilistic model can be written as

$$\mu_i \sim \mathcal{N}(0, \Sigma), \quad z_i|\mu_i \sim \mathcal{N}(\mu_i, I).$$

Since the prior distribution and the conditional distribution are Gaussians, we have the marginal distribution by integrating out μ_i,

$$z_i \sim \mathcal{N}(0, \Sigma + I). \tag{3}$$

and the posterior distribution

$$\mu_i|z_i \sim \mathcal{N}((\Sigma^{-1} + I)^{-1} z_i, (\Sigma^{-1} + I)^{-1}) \tag{4}$$

One can use method of moments (MoM) estimator or MLE to estimate Σ based on (3). Here we simply use MoM: $\hat{\Sigma} = \frac{1}{n}\sum_i z_i z_i^T - I$.

(b) For a mis-specified model, one may consider

$$\mu_i \sim \pi\mathcal{N}(0, \Sigma) + (1-\pi)\mathcal{N}(0, I),$$

where $\Sigma = \begin{pmatrix} \sigma_A^2 & \rho\sigma_A\sigma_B \\ \rho\sigma_A\sigma_B & \sigma_B^2 \end{pmatrix}$. The proportion π can vary in $\{0.3, 0.6, 0.9\}$.

```
library(mvtnorm)
library(ggplot2)

bvJSE <- function(Z){
  # Z is n by 2 matrix
  ZZ <- t(Z)%*%Z
  Sig <- ZZ/n - diag(2)
  mu <-Z%*% solve(solve(Sig)+diag(2))
  return(list(mu=mu,Sigma=Sig))
}

JSE <- function(z){
  n <- length(z)
  tmp <- (n-2)/sum(z^2)
  mu <- (1-tmp)%*%z
  return(mu)
}


################################## simulation ###################################

n <- 500
sigA <- 1
```

```
sigB <- 2
rho_all <- c(0,0.4,0.6,0.8,1)


nrep <- 100

err_muA <- err_muB <- matrix(0,nrep,2)

out <- data.frame()

for(j in 1:length(rho_all)){
  rho <- rho_all[j]
  Sigma <- matrix(c(sigA,rho*sqrt(sigA*sigB),rho*sqrt(sigA*sigB),sigB),2,2)
  for(i in 1:nrep){
    mu <- rmvnorm(n,rep(0,2),Sigma)
    Z <- matrix(rnorm(2*n,c(mu),1),n,2)

    fit_bv <- bvJSE(Z)
    err_muA[i,1] <- mean((fit_bv$mu[,1]-mu[,1])^2)
    err_muB[i,1] <- mean((fit_bv$mu[,2]-mu[,2])^2)

    fit_JSE <- apply(Z,2,JSE)
    err_muA[i,2] <- mean((fit_JSE[,1]-mu[,1])^2)
    err_muB[i,2] <- mean((fit_JSE[,2]-mu[,2])^2)

    out <- rbind(out,data.frame(error=mean((fit_bv$mu[,1]-mu[,1])^2),
                                mu="mu_A",method="bv-JSE",
                                correlation=paste0("correaltion=",rho)))
    out <- rbind(out,data.frame(error=mean((fit_bv$mu[,2]-mu[,2])^2),
                                mu="mu_B",method="bv-JSE",
                                correlation=paste0("correaltion=",rho)))
    out <- rbind(out,data.frame(error=mean((fit_JSE[,1]-mu[,1])^2),
                                mu="mu_A",method="JSE",
                                correlation=paste0("correaltion=",rho)))
    out <- rbind(out,data.frame(error=mean((fit_JSE[,2]-mu[,2])^2),
                                mu="mu_B",method="JSE",
                                correlation=paste0("correaltion=",rho)))

  }
}

P_simu <- ggplot(out,aes(x=method,y=error)) + geom_boxplot() + facet_grid(mu~correlation)
P_simu
################################## mis-specified model ###################################
n <- 500
pcor_all <- c(0.3,0.6,0.9)
sigA <- 1
sigB <- 2
rho_all <- c(0,0.4,0.6,0.8,1)

nrep <- 100

err_muA <- err_muB <- matrix(0,nrep,2)
```

```
est_cor <- rep(0,nrep)

out <- data.frame()

for(k in 1:length(pcor_all)){
  pcor <- pcor_all[k]
  n1 <- n*pcor
  n2 <- n-n1
  for(j in 1:length(rho_all)){
    rho <- rho_all[j]
    Sigma <- matrix(c(sigA,rho*sqrt(sigA*sigB),rho*sqrt(sigA*sigB),sigB),2,2)

    for(i in 1:nrep){
      mu <- rmvnorm(n1,rep(0,2),Sigma)
      mu <- rbind(mu,rmvnorm(n2,rep(0,2),diag(diag(Sigma))))
      Z <- matrix(rnorm(2*n,c(mu),1),n,2)

      fit_bv <- bvJSE(Z)
      err_muA[i,1] <- mean((fit_bv$mu[,1]-mu[,1])^2)
      err_muB[i,1] <- mean((fit_bv$mu[,2]-mu[,2])^2)

      est_cor[i] <- with(fit_bv,Sigma[1,2]/sqrt(Sigma[1,1]*Sigma[2,2]))

      fit_JSE <- apply(Z,2,JSE)
      err_muA[i,2] <- mean((fit_JSE[,1]-mu[,1])^2)
      err_muB[i,2] <- mean((fit_JSE[,2]-mu[,2])^2)

      out <- rbind(out,data.frame(error=mean((fit_bv$mu[,1]-mu[,1])^2),
                                  mu="mu_A",method="bv-JSE",
                                  correlation=paste0("correaltion=",rho),
                                  pcor=paste0("n correlated=",n1)))
      out <- rbind(out,data.frame(error=mean((fit_bv$mu[,2]-mu[,2])^2),
                                  mu="mu_B",method="bv-JSE",
                                  correlation=paste0("correaltion=",rho),
                                  pcor=paste0("n correlated=",n1)))
      out <- rbind(out,data.frame(error=mean((fit_JSE[,1]-mu[,1])^2),
                                  mu="mu_A",method="JSE",
                                  correlation=paste0("correaltion=",rho),
                                  pcor=paste0("n correlated=",n1)))
      out <- rbind(out,data.frame(error=mean((fit_JSE[,2]-mu[,2])^2),
                                  mu="mu_B",method="JSE",
                                  correlation=paste0("correaltion=",rho),
                                  pcor=paste0("n correlated=",n1)))
    }
  }
}

P_misspecA <- ggplot(subset(out,mu=="mu_A"),aes(x=method,y=error)) +
  geom_boxplot() + facet_grid(pcor~correlation) + ggtitle("error of mu_A")
P_misspecA

P_misspecB <- ggplot(subset(out,mu=="mu_B"),aes(x=method,y=error)) +
  geom_boxplot() + facet_grid(pcor~correlation) + ggtitle("error of mu_B")
```

P_misspecB

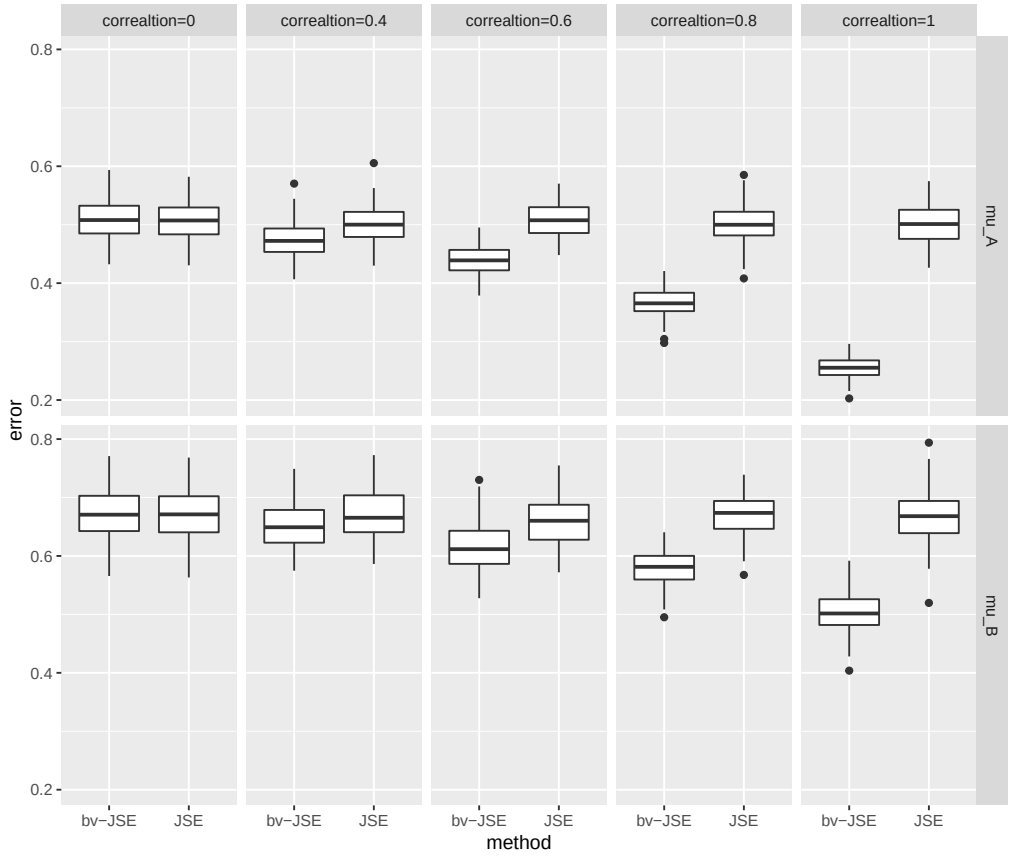

Figure 3: Results of $\mathbb{E}[\|\mu_A - \hat{\mu}_A\|^2]$ and $\mathbb{E}[\|\mu_B - \hat{\mu}_B\|^2]$ with correct model specification.

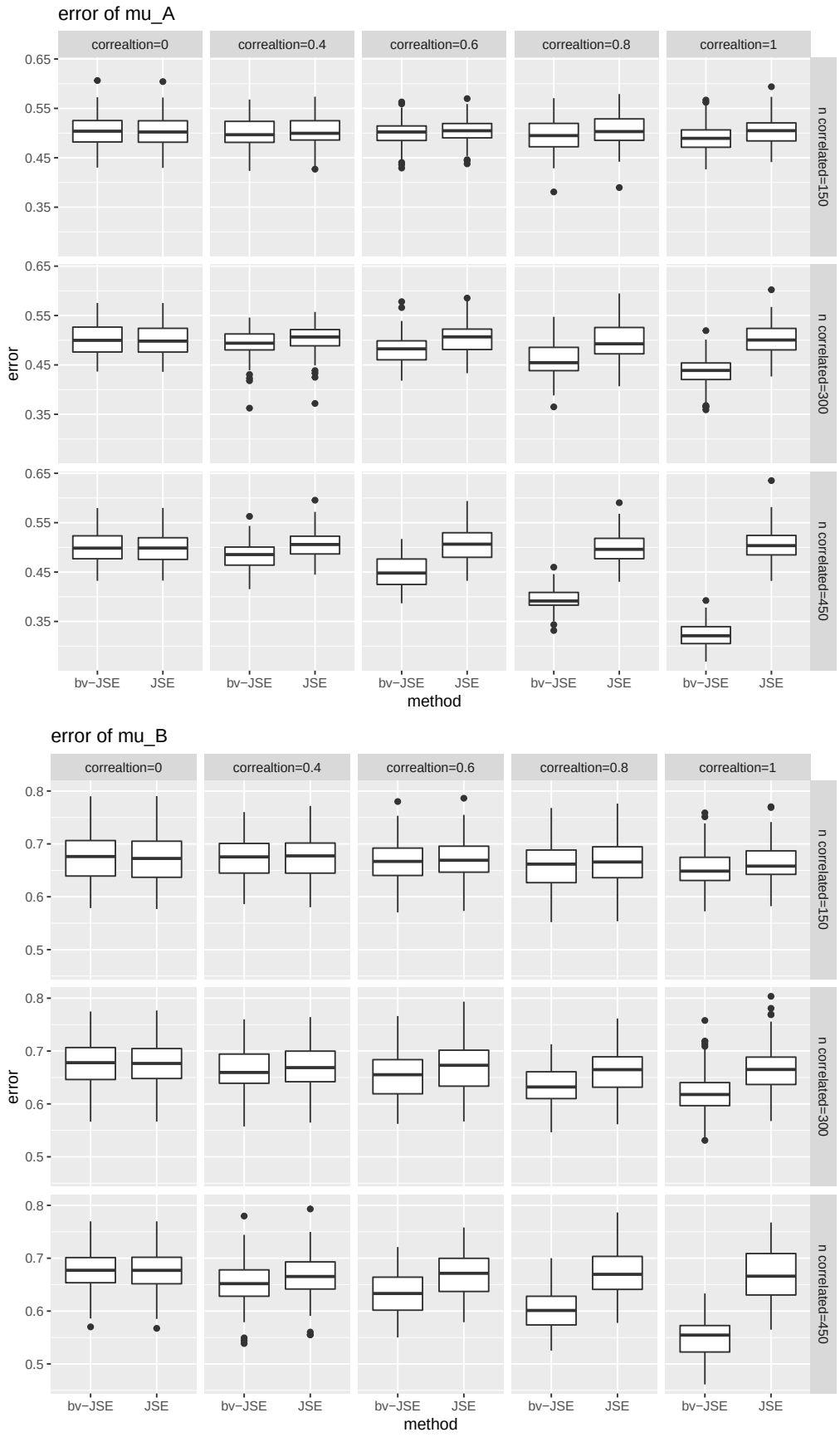

Figure 4: Results of $\mathbb{E}[\|\mu_A - \hat{\mu}_A\|^2]$ (top panel) and $\mathbb{E}[\|\mu_B - \hat{\mu}_B\|^2]$ (bottom panel) with model mis-specification.