Regression Analysis HW6

Tuorui Peng, tuoruipeng2028@u.northwestern.edu

Question 3.1

(a)

We can simply define $Z = XV_rV_r' \in \mathbb{R}^{n\times r}$, $r \in [d]$. Then we can see that b is just the OLS estimator of Y over Z, so the dof is simply

$$\mathrm{dof}(\hat{f}_{\mathrm{pcr}}) = tr(Z(Z'Z)^{-1}Z') + 1 = r + 1$$

(b)

We know that ridge regression estimator is

$$\hat{\beta}_\lambda = (X'X + \lambda I)^{-1}X'Y$$

so

$$\begin{aligned}
\mathrm{dof}(\hat{f}_{\mathrm{ridge}}) =& \frac{1}{\sigma^2}\sum_{i=1}^n cov(\hat{Y}_i, Y_i) \\
=& \frac{1}{\sigma^2}\mathbb{E}\left[(Y - \mathbb{E}[Y])'\hat{Y}\right] \\
=& \frac{1}{\sigma^2}\mathbb{E}\left[(Y - \mathbb{E}[Y])'X(X'X + \lambda I)^{-1}X'Y\right] \\
=& \frac{1}{\sigma^2}\mathbb{E}\left[\varepsilon'X(X'X + \lambda I)^{-1}X'\varepsilon'\right] \\
=& tr(X(X'X + \lambda I)^{-1}X')
\end{aligned}$$

Question 3.2

(a)

The Kernel is just mapping the original data points to a new spaces, where we expect the 'structure' of the data is better captured.

(b)

Plug in the kernel expression $h(\cdot) = \sum_{i=1}^n \alpha_i k(x, x_i)$, we have

$$\begin{aligned}
\hat{\alpha} =& \arg\min_{\alpha} \sum_{i=1}^n (y_i - \sum_{j=1}^n \alpha_j k(x_i, x_j))^2 + \lambda \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j k(x_i, x_j) \\
=& \arg\min_{\alpha} (Y - G\alpha)'(Y - G\alpha) + \lambda \alpha' G \alpha \\
:=& \arg\min_{\alpha} \mathcal{L}(\alpha)
\end{aligned}$$

the minimizer satisties

$$\begin{aligned}
0 = \frac{\partial \mathcal{L}}{\partial \alpha} =& -2G'Y + 2G'G\alpha + 2\lambda G\alpha \\
\Rightarrow \hat{\alpha} =& (G + \lambda I)^{-1} Y
\end{aligned}$$

Yes, because the regularizer λI will help keep the matrix $G + \lambda I$ invertible.

(c)

With model $y = f(x) + \varepsilon$, $\varepsilon \sim (0, \sigma^2)$, we have

$$\begin{aligned}
\text{dof}(\hat{f}_{\text{KRR}}) =& \frac{1}{\sigma^2} \sum_{i=1}^n cov(\hat{Y}_i, Y_i) \\
=& \frac{1}{\sigma^2} tr\big(cov(G(G + \lambda I)^{-1}(f(X) + \varepsilon), f(X) + \varepsilon)\big) \\
=& \frac{1}{\sigma^2} tr\big(G(G + \lambda I)^{-1} \sigma^2\big) = tr\big(G(G + \lambda I)^{-1}\big)
\end{aligned}$$

(d)

For RBF Kernel $k(\cdot, \cdot) = \exp(-\| \cdot - \cdot \|_2^2 / 2\tau^2)$, we have

- **(i)** for $\tau^2 \to \infty$, we have $k(x_i, x_j) \to 1$, $\forall i, j \in [n]$, so

$$\begin{aligned}
\lim_{\tau^2 \to \infty} \text{dof}(\hat{f}_{\text{KRR}}) =& \lim_{\tau^2 \to \infty} tr(G(G + \lambda I)^{-1}) = tr(\mathbb{1}\mathbb{1}'(\mathbb{1}\mathbb{1}' + \lambda I)^{-1}) \\
\overset{\text{SMW}}{=}& tr\Big(\big(\frac{1}{\lambda}(I - \frac{\mathbb{1}\mathbb{1}'}{n + \lambda})\mathbb{1}\mathbb{1}'\big)\Big) = \frac{1}{\lambda}(1 - \frac{n}{n + \lambda}) tr(\mathbb{1}\mathbb{1}') \\
=& \frac{n}{n + \lambda}
\end{aligned}$$

- **(ii)** for $\tau^2 \to 0$, we have $k(x_i, x_j) \to \delta_{ij}$, $\forall i, j \in [n]$, so

$$\lim_{\tau^2 \to 0} \text{dof}(\hat{f}_{\text{KRR}}) = \lim_{\tau^2 \to 0} tr(I(G + \lambda I)^{-1}) = tr((I + \lambda I)^{-1}) = \frac{n}{1 + \lambda}$$

  if $\lambda \approx 0$ and $\tau \approx 0$ we have $\text{dof}(\hat{f}_{\text{KRR}}) \approx 1$, Here the 1 degree of freedom is just the intercept or 'mean value' term, so the model is just a constant model, not capturing any structure of the data.

And in this case, the estimator is

$$\hat{y}_i = \sum_{j=1}^{n} \delta_{ij}\alpha_j = \alpha_i = \frac{y_i}{1+\lambda} \to y_i$$

which is just a 'local' estimate, each $\hat{y}_i$ is almost its observed value y_i.

(e)

We have

$$\begin{aligned}
\mathbf{f} - \mathbb{E}\left[\hat{Y}\right] =& \mathbf{f} - \mathbb{E}\left[G(G+\lambda I)^{-1}(\mathbf{f}+\varepsilon)\right] \\
=& (I - G(G+\lambda I)^{-1})\mathbf{f} \\
\overset{\text{SMW}}{=}& (I - G(G^{-1} - G^{-1}(\frac{1}{\lambda}I + G^{-1})^{-1}G^{-1}))\mathbf{f} \\
=& \lambda(I + \lambda G^{-1})^{-1}G^{-1}\mathbf{f} \\
=& \lambda G^{-1}\mathbf{f} + O(\lambda^2)
\end{aligned}$$

(f)

We have

$$\begin{aligned}
\mathbb{E}\left[\text{RSS}\right] =& \mathbb{E}\left[\|\hat{Y} - Y\|_2^2\right] = \mathbb{E}\left[(\mathbf{f}+\varepsilon)'(I-H_\lambda)'(I-H_\lambda)(\mathbf{f}+\varepsilon)\right] \\
=& \mathbf{f}(I-H_\lambda)'(\mathbf{f} - \mathbb{E}\left[\hat{Y}\right]) + \mathbb{E}\left[\varepsilon'(I-H_\lambda)'(I-H_\lambda)\varepsilon\right] \\
=& \mathbf{f}(I-H_\lambda)'\lambda G^{-1}\mathbf{f} + \sigma^2 tr((I-H_\lambda)'(I-H_\lambda)) + O(\lambda^2) \\
=& \lambda\mathbf{f}'G^{-1}(I - (I+\lambda G^{-1})^{-1})\mathbf{f} + \sigma^2 tr((I-H_\lambda)'(I-H_\lambda)) + O(\lambda^2) \\
=& \lambda\mathbf{f}'G^{-1}(\lambda G^{-1} + O(\lambda^2))\mathbf{f} + \sigma^2 tr((I-H_\lambda)^2) + O(\lambda^2) \\
=& \sigma^2 tr((I-H_\lambda)) \\
=& \sigma^2(n - 2tr(H_\lambda) + tr(H_\lambda^2)) + O(\lambda^2) \\
=& \sigma^2(n - 2\cdot \text{dof}(\hat{f}) + tr(H_\lambda^2)) + O(\lambda^2)
\end{aligned}$$

In this way, if we use $\hat{\sigma}^2 = \dfrac{1}{n - 2\cdot\text{dof}(\hat{f}) + tr(H_\lambda^2)}\|\hat{Y} - Y\|_2^2$, then we have

$$\mathbb{E}\left[\hat{\sigma}^2\right] = \frac{1}{n - 2\cdot\text{dof}(\hat{f}) + tr(H_\lambda^2)}\mathbb{E}\left[\|\hat{Y} - Y\|_2^2\right] = \sigma^2 + \frac{O(\lambda^2)}{n - 2\cdot\text{dof}(\hat{f}) + tr(H_\lambda^2)}$$

Question 3.3

(a)(b)

```r
library('tidyverse')
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.1.1     v forcats 0.5.1
```

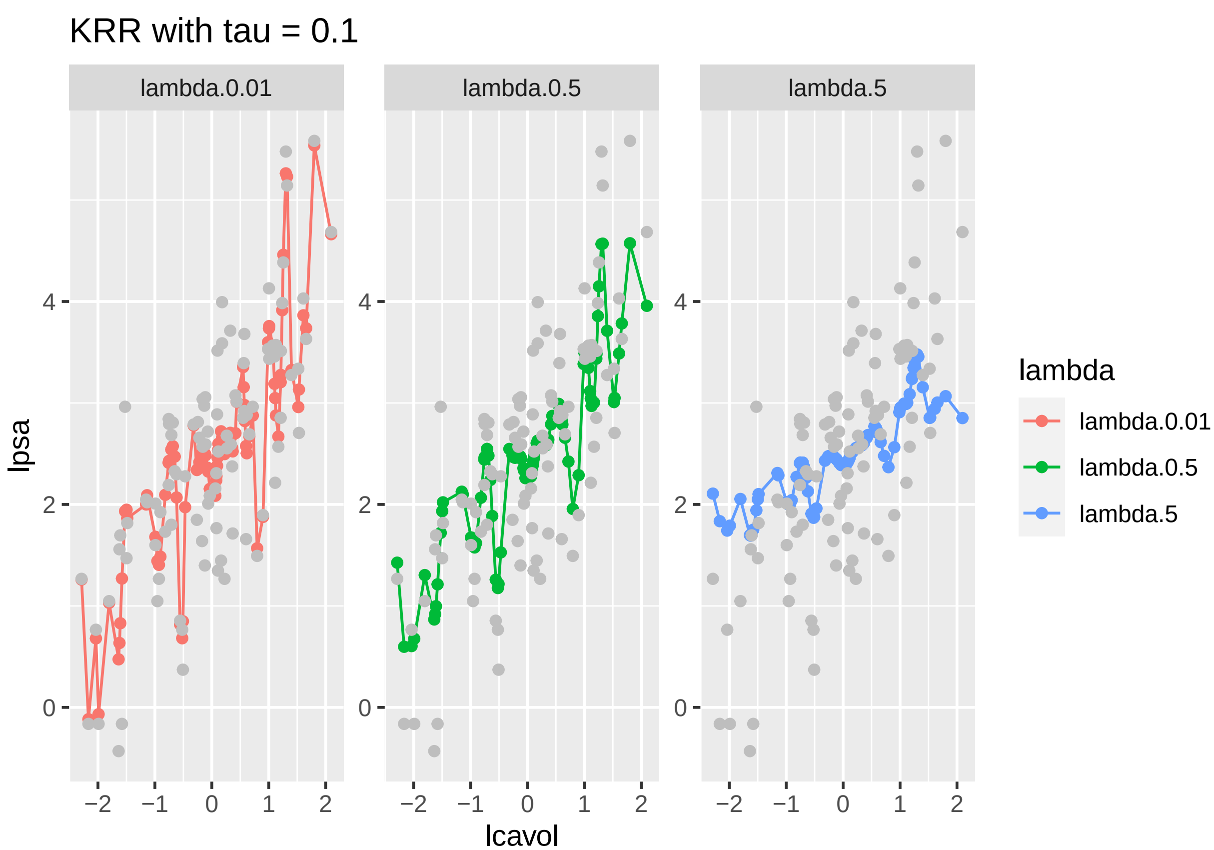
```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
lprostate <- read.csv("lprostate.dat", sep = "\t")

predictKRR <- function(X,Z,alpha,tau,offset){
    ## X: n by d matrix, training data
    ## Z: m by d matrix, data to be predicted
    ## alpha: n by 1 vector, KRR estimator
    ## tau: scalar in RBF function exp(-||x-y||^2/2tau^2)
    ## offset: scalar, intercept
    dist_xzfull <- dist(rbind(X, Z), method = 'euclidean')
    dist_xz <- as.matrix(dist_xzfull[ (nrow(X)+1):nrow(dist_xzfull),1:nrow(X)])
    K <- exp(-dist_xz^2 / (2 * tau^2))
    return(K %*% alpha + offset)
}
fitKRR <- function(X,y,lambda,tau){
    ## X: n by d matrix, training data
    ## y: n by 1 vector, training response
    ## lambda: scalar, regularization parameter
    ## tau: scalar in RBF function exp(-||x-y||^2/2tau^2)
    n <- nrow(X)
    centered_y <- y - mean(y)
    K <- exp(-as.matrix(dist(X, method = 'euclidean'))^2 / (2 * tau^2))
    alpha <- solve(K + lambda * diag(n)) %*% centered_y
    yMean <- K %*% alpha + mean(y)
    return( list(alpha = alpha, yMean = yMean) )
}
```

(c)

From the plot we can see that the model with $\tau = 0.1$ seems to be too 'local', we observe that the model with $\lambda = 0.01$ could fit the overall trend but has significant overfitting, while the model with $\lambda = 5$ is too 'smooth' and cannot capture the trend.

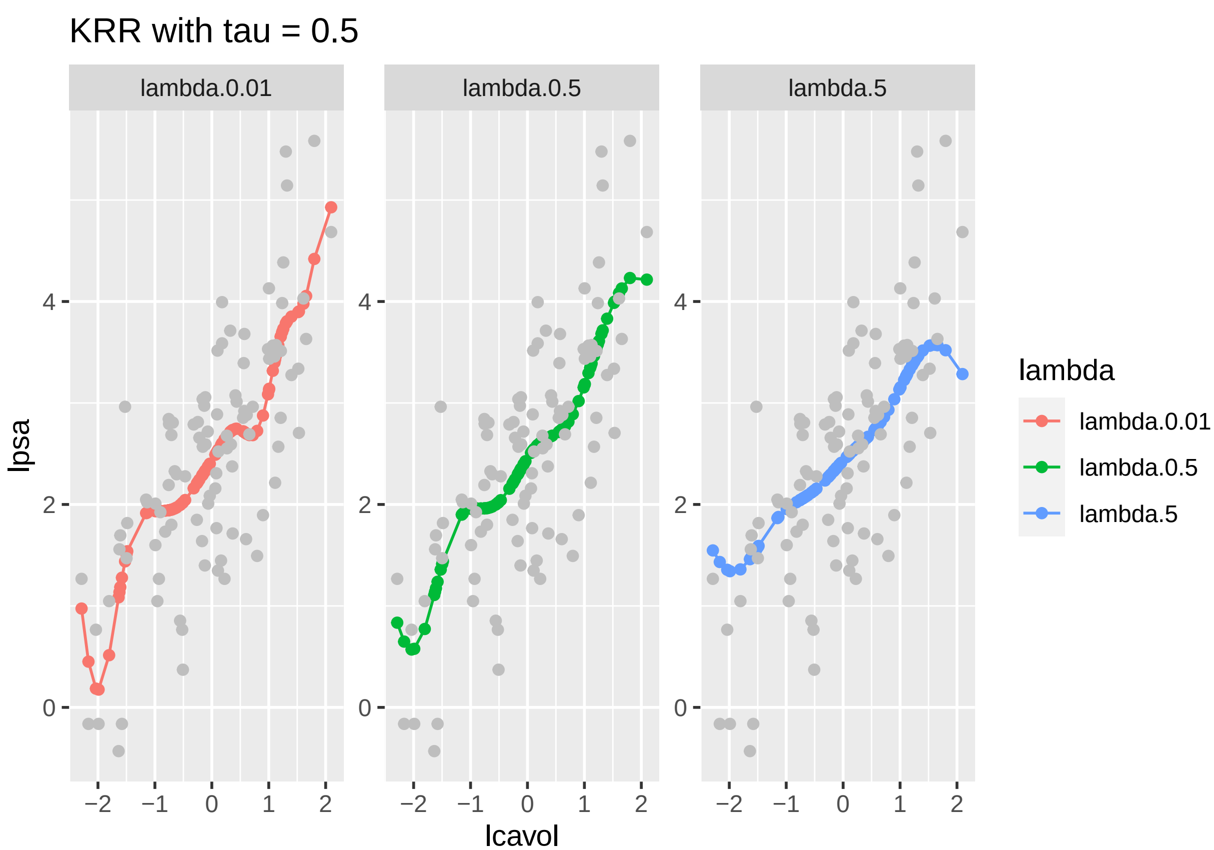
```r
# fit KRR on lprostate data with tau = 0.1
X <- lprostate[, 'lcavol'] %>% scale() %>% as.matrix()
y <- lprostate[, 'lpsa']
tau <- 0.1
lambdas <- c(0.01, 0.5, 5)
KRR_tau_0.1 <- data.frame(x = X, y = y, lambda.0.01 = NA, lambda.0.5 = NA, lambda.5 = NA)
for(lambda in lambdas){
    fit <- fitKRR(X, y, lambda, tau)
    KRR_tau_0.1[, paste0('lambda.', lambda)] <- fit$yMean
}
# 3 plot facets, showing y and yhat against x, for each lambda
# add legend to the bottom, declaring which color is observed data and which is predicted
KRR_tau_0.1 %>% gather(key = "lambda", value = "yhat", -x, -y) %>% ggplot(aes(x = x, y = yhat, color
```



(d)

Plot for $\tau = 0.5$ seems to be quite good, models with $\lambda = 0.01$ and $\lambda = 0.5$ could fit quite well, being smooth enough and capturing the trend.

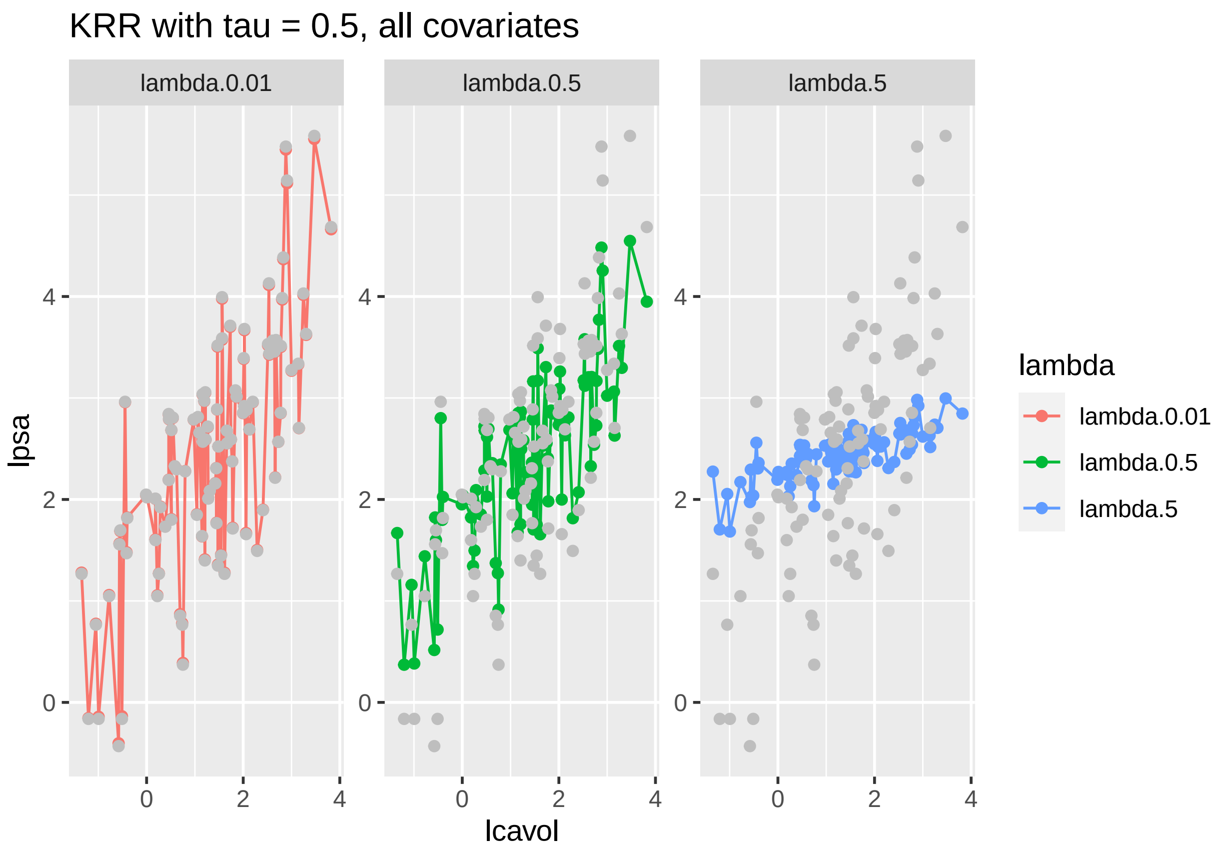
```r
# fit KRR on lprostate data with tau = 0.5
X <- lprostate[, 'lcavol'] %>% scale() %>% as.matrix()
y <- lprostate[, 'lpsa']
tau <- 0.5
lambdas <- c(0.01, 0.5, 5)
KRR_tau_0.5 <- data.frame(x = X, y = y, lambda.0.01 = NA, lambda.0.5 = NA, lambda.5 = NA)
for(lambda in lambdas){
    fit <- fitKRR(X, y, lambda, tau)
    KRR_tau_0.5[, paste0('lambda.', lambda)] <- fit$yMean
}
# 3 plot facets, showing y and yhat against x, for each lambda
# add legend to the bottom, declaring which color is observed data and which is predicted
KRR_tau_0.5 %>% gather(key = "lambda", value = "yhat", -x, -y) %>% ggplot(aes(x = x, y = yhat, color
```



(e)

Plot for $\tau = 0.5$ with all covariates involve. This time the prediction seems to be less satisfactory, the model with $\lambda = 0.01$ is too 'local' and the model with $\lambda = 5$ is too close to mean, perhaps $\lambda = 0.5$ model is better, because we can't see the plotting projection on other dimensions.

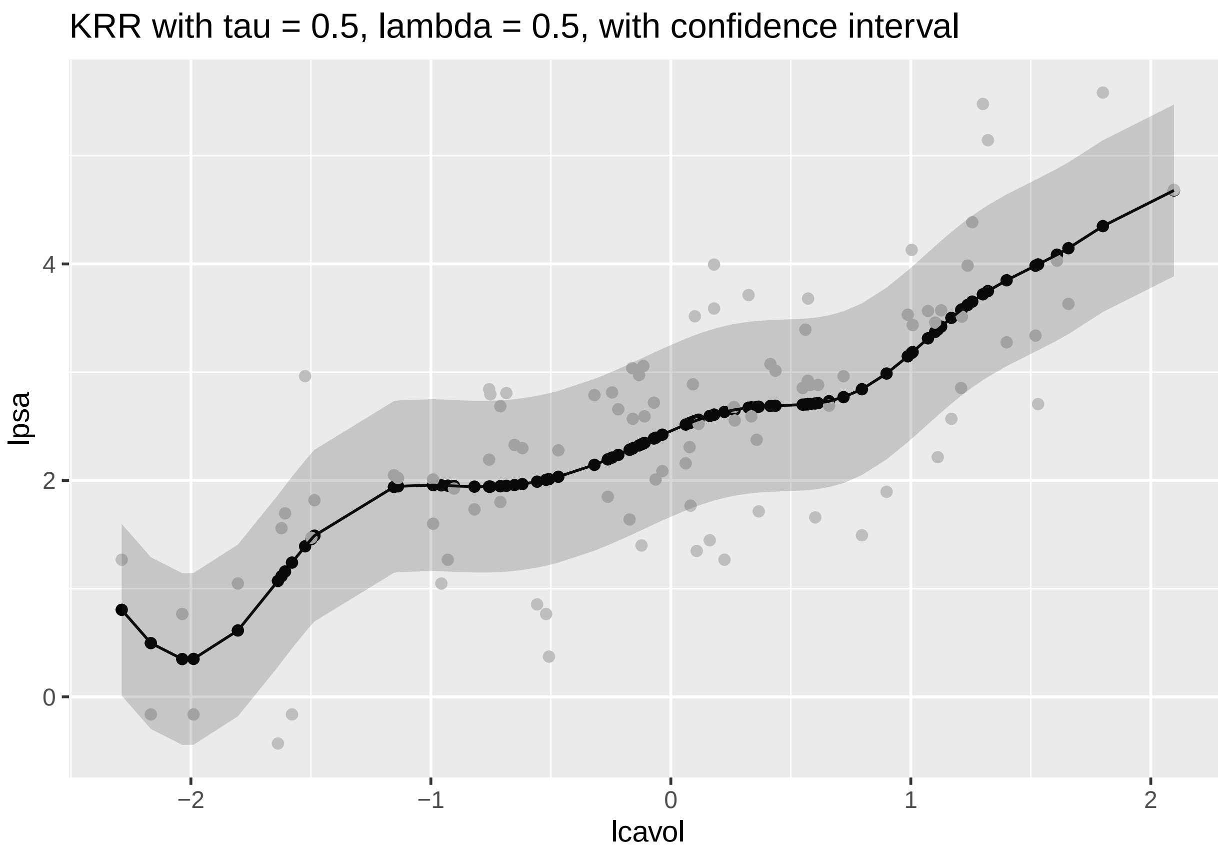
```
# fit KRR on lprostate data with tau = 0.5
X <- lprostate[, c('lcavol', 'lweight', 'age', 'lbph', 'svi', 'lcp', 'gleason', 'pgg45')] %>% scale()
y <- lprostate[, 'lpsa']
tau <- 0.5
lambdas <- c(0.01, 0.5, 5)
KRR_tau_0.5_all <- data.frame(x = lprostate$lcavol, y = y, lambda.0.01 = NA, lambda.0.5 = NA, lambda.
for(lambda in lambdas){
    fit <- fitKRR(X, y, lambda, tau)
    KRR_tau_0.5_all[, paste0('lambda.', lambda)] <- fit$yMean
}
# 3 plot facets, showing y and yhat against x, for each lambda
# add legend to the bottom, declaring which color is observed data and which is predicted
KRR_tau_0.5_all %>% gather(key = "lambda", value = "yhat", -x, -y) %>% ggplot(aes(x = x, y = yhat, co
```



(f)

```
# fit KRR on lprostate data with tau = 0.5, lambda = 0.1 to get variance estimator
X <- lprostate[, c('lcavol', 'lweight', 'age', 'lbph', 'svi', 'lcp', 'gleason', 'pgg45')] %>% scale()
y <- lprostate[, 'lpsa']
tau <- 0.5
```

```
lambdas <- 0.1
KRR_tau_0.5_all_getvar <- data.frame(x = lprostate$lcavol, y = y, lambda.0.1 = NA)
for(lambda in lambdas){
    fit <- fitKRR(X, y, lambda, tau)
    KRR_tau_0.5_all_getvar[, paste0('lambda.', lambda)] <- fit$yMean
}
## get variance estimator
yhat <- KRR_tau_0.5_all_getvar$lambda.0.1
n <- length(yhat)
K <- exp(-as.matrix(dist(X, method = 'euclidean'))^2 / (2 * tau^2))
H <- K %*% solve(K + lambda * diag(n))
sigma2_hat <- 1 / (n - 2 * sum(diag(H)) + sum(diag( t(H) %*% H ))) * (sum((y - yhat)^2) )


# fit KRR on lprostate data with tau = 0.5, lambda = 0.1
X <- lprostate[, 'lcavol'] %>% scale() %>% as.matrix()
y <- lprostate[, 'lpsa']
tau <- 0.5
lambdas <- 0.1
KRR_tau_0.5band <- data.frame(x = X, y = y, lambda.0.1 = NA)
for(lambda in lambdas){
    fit <- fitKRR(X, y, lambda, tau)
    KRR_tau_0.5band[, paste0('lambda.', lambda)] <- fit$yMean
}
KRR_tau_0.5band$upper <- KRR_tau_0.5band$lambda.0.1 + sqrt(sigma2_hat)
KRR_tau_0.5band$lower <- KRR_tau_0.5band$lambda.0.1 - sqrt(sigma2_hat)
# plot showing y and yhat against x, with confidence interval using variance estimator

KRR_tau_0.5band %>% ggplot(aes(x = x, y = lambda.0.1)) + geom_point() + geom_point(aes(x = x, y = y),
```



Question 3.4

(a)

We have

$$\widehat{\beta}_\lambda := \arg\min_b \|Xb - y\|_2^2 + b'V\Lambda V'b := \arg\min_b \mathcal{L}(b)$$

the minimizer satisfy

$$\begin{aligned} 0 = \frac{\partial \mathcal{L}}{\partial b} =& 2X'(Xb - y) + 2V\Lambda V'b \\ \Rightarrow \widehat{\beta}_\lambda =& (X'X + V\Lambda V')^{-1}X'y = (V\Gamma^2V' + V\Lambda V')^{-1}V\Gamma U'y = V(\Gamma^2 + \Lambda)^{-1}\Gamma U'y = V\Gamma(\Gamma^2 + \Lambda)^{-1}U'y \end{aligned}$$

similarly, we have

$$H_\lambda = X(X'X + V\Lambda V')^{-1}X' = U\Gamma(\Gamma^2 + \Lambda)^{-1}\Gamma U' = U\Gamma^2(\Gamma^2 + \Lambda)^{-1}U'$$

(b)

Note that with model $Y_i = f(x_i) + \varepsilon_i$, $\varepsilon_i \sim (0, \sigma^2)$ we have

$$\mathbb{E}\left[(\widehat{Y}_\lambda - f(x)) \cdot (f(x) - Y)\right] = cov(\widehat{Y}_\lambda, Y)$$

thus

$$\begin{aligned}\frac{1}{n}\mathbb{E}\left[\text{RSS}\right] =& \frac{1}{n}\mathbb{E}\left[\|\hat{Y}_\lambda - Y\|_2^2\right]\\ =& \frac{1}{n}\sum_{i=1}^n \mathbb{E}\left[(\hat{Y}_{\lambda,i} - Y_i)^2\right] = \frac{1}{n}\sum_{i=1}^n \mathbb{E}\left[(\hat{Y}_{\lambda,i} - f(x_i) + f(x_i) - Y_i)^2\right]\\ =& \frac{1}{n}\left[\mathbb{E}\left[(\hat{Y}_{\lambda,i} - f(x_i))^2\right] + \mathbb{E}\left[(f(x_i) - Y_i)^2\right] - 2\mathbb{E}\left[(\hat{Y}_{\lambda,i} - f(x_i))\cdot(f(x_i) - Y_i)\right]\right]\\ =& R_{\text{in}}(\hat{\beta}_\lambda) + \sigma^2 - \frac{2}{n}\sum_{i=1}^n cov(\hat{Y}_{\lambda,i}, Y_i)\end{aligned}$$

in which note that

$$\begin{aligned}cov(\hat{Y}_\lambda, Y) =& \mathbb{E}\left[(\hat{Y}_\lambda - f(x))\cdot(f(x) - Y)\right]\\ =& \mathbb{E}\left[\hat{Y}_\lambda\cdot(f(x) - Y)\right]\\ =& \mathbb{E}\left[(H_\lambda Y)\cdot(f(x) - Y)\right]\\ =& \mathbb{E}\left[\varepsilon' H_\lambda \varepsilon\right]\\ =& \sigma^2 tr(H_\lambda)\end{aligned}$$

which gives

$$\frac{1}{n}\mathbb{E}\left[\|\hat{Y}_\lambda - Y\|_2^2\right] + \frac{2\sigma^2}{n}tr(H_\lambda) = R_{\text{in}}(\hat{\beta}_\lambda) + \sigma^2$$

(c)

We have

$$\begin{aligned}r(\lambda) =& \frac{1}{n}\|\hat{Y}_\lambda - Y\|_2^2 + \frac{2\hat{\sigma}^2}{n}tr(H_\lambda)\\ =& \frac{1}{n}Y'(I - H_\lambda)'(I - H_\lambda)Y + \frac{2\hat{\sigma}^2}{n}tr(H_\lambda)\\ =& \frac{1}{n}Y'(I - U\Gamma^2(\Gamma^2+\Lambda)^{-1}U')'(I - U\Gamma^2(\Gamma^2+\Lambda)^{-1}U')Y + \frac{2\hat{\sigma}^2}{n}tr(U\Gamma^2(\Gamma^2+\Lambda)^{-1}\Gamma U')\\ =& \frac{1}{n}Y'(UU' + U_\perp U_\perp' - U\Gamma^2(\Gamma^2+\Lambda)^{-1}U')^2 Y + \frac{2\hat{\sigma}^2}{n}tr(U\Gamma^2(\Gamma^2+\Lambda)^{-1}\Gamma U')\\ =& \frac{1}{n}Y'U(I - \Gamma^2(\Gamma^2+\Lambda)^{-1})U'Y + \frac{1}{n}Y'U_\perp U_\perp' Y + \frac{2\hat{\sigma}^2}{n}tr(U\Gamma^2(\Gamma^2+\Lambda)^{-1}\Gamma U')\\ =& \frac{1}{n}\sum_{j=1}^d\left[\frac{\lambda_j^2}{(\gamma_j^2+\lambda_j)^2}(u_j'Y)^2 + 2\hat{\sigma}^2\frac{\gamma_j^2}{\gamma_j^2+\lambda_j}\right] + \frac{1}{n}\|U_\perp' Y\|_2^2\end{aligned}$$

and take derivative $\dfrac{\partial}{\partial\lambda_j}$ to obtain

$$\begin{aligned}\frac{n}{2}\frac{\partial}{\partial\lambda_j}r(\lambda) =& \frac{1}{2}\frac{\partial}{\partial\lambda_j}\sum_{j=1}^d\left[\frac{\lambda_j^2}{(\gamma_j^2+\lambda_j)^2}(u_j'Y)^2 + 2\hat{\sigma}^2\frac{\gamma_j^2}{\gamma_j^2+\lambda_j}\right]\\ =& \frac{\lambda_j}{\gamma_j^2+\lambda_j}\cdot\frac{\gamma_j^2}{(\gamma_j^2+\lambda_j)^2}\cdot(u_j'Y)^2 - \hat{\sigma}^2\frac{\gamma_j^2}{(\gamma_j^2+\lambda_j)^2}\\ =& \frac{\gamma_j^2}{(\gamma_j^2+\lambda_j)^2}\cdot\left[\frac{\lambda_j}{\gamma_j^2+\lambda_j}\cdot(u_j'Y)^2 - \hat{\sigma}^2\right]\end{aligned}$$

(d)

First for $\lambda \geq 0$, we have

$$\frac{\partial}{\partial \lambda_j} r(\lambda) \propto \frac{\lambda_j}{\gamma_j^2 + \lambda_j} \cdot (u_j' Y)^2 - \hat{\sigma}^2 < (u_j' Y)^2 - \hat{\sigma}^2 < 0, \quad \text{iff } \hat{\sigma}^2 \geq (u_j' Y)^2$$

and for $\hat{\sigma}^2 < (u_j' Y)^2$, we have

$$0 = \frac{\partial}{\partial \lambda_j} r(\lambda) \propto \frac{\lambda_j}{\gamma_j^2 + \lambda_j} \cdot (u_j' Y)^2 - \hat{\sigma}^2 \Rightarrow \lambda_j^* = \frac{\hat{\sigma}^2 \gamma_j^2}{(u_j' Y)^2 - \hat{\sigma}^2}$$

to summarize, we have

$$\lambda^* = \begin{cases} +\infty, & \text{if } \hat{\sigma}^2 \geq (u_j' Y)^2 \\ \dfrac{\hat{\sigma}^2 \gamma_j^2}{(u_j' Y)^2 - \hat{\sigma}^2}, & \text{if } \hat{\sigma}^2 < (u_j' Y)^2 \end{cases}$$

Intuitively, if $\hat{\sigma}^2$ is too large, then any penalization will not be enough to reduce in-sample risk, because the signal of the fitted model is dominated by noise. On the other hand, if $\hat{\sigma}^2$ is small, then we can use a proper penalization to reduce in-sample risk, i.e. increase signal-to-noise ratio.

(e)

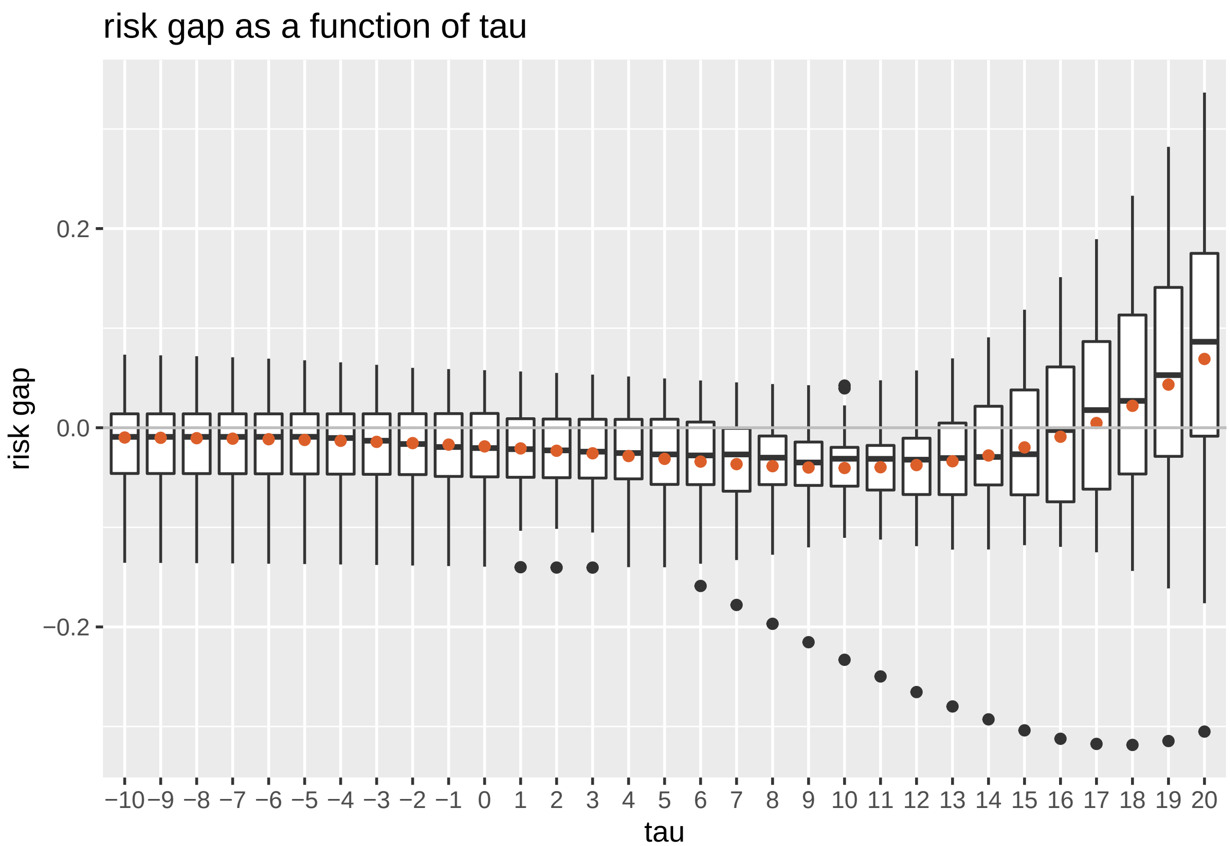
Simulation on `lprostate.dat`

```r
## ridge-prostatde-dataprep.r
library(tidyverse);
prostate.data = tibble(read.csv("lprostate.dat", sep = "\t", header=TRUE));

construct.train.and.test <- function(dataset, response = "", split.prop = .6) {
    if (all(names(dataset) != response)) {
        stop("Dataset does not contain the given response")
    }
    total.n = nrow(dataset)
    permutation = sample(1:total.n, total.n)
    train.size = as.integer(split.prop * total.n)
    train.data = dataset[permutation[1:train.size], ]
    test.data = dataset[permutation[(train.size+1):total.n], ]

    X.train = train.data[, !names(dataset) %in% c(response)]
    y.train = train.data[, response]
    X.test = test.data[, !names(dataset) %in% c(response)]
    y.test = test.data[, response]

    y.train = scale(y.train, scale = F)
```

```
    X.train = scale(X.train)
    y.test = y.test - attr(y.train, "scaled:center")
    X.test = sapply(1:dim(X.test)[2],
                    function(i) {
                        ((X.test[,i] - attr(X.train, "scaled:center")[i])
                            / attr(X.train, "scaled:scale")[i])
                    })

    train.data[, !names(dataset) %in% c(response)] = X.train
    train.data[, response] = y.train
    test.data[, !names(dataset) %in% c(response)] = X.test
    test.data[, response] = y.test

    return(list(train = tibble(train.data),
                test = tibble(test.data)))
}

## Remove columns and save our target name
prostate.data = prostate.data[! names(prostate.data) == "row.names"];

## Simulation
tau <- 10^((-10:20)/10)
N <- 25

risk.gap.df <- matrix(NA, nrow = length(tau), ncol = N)
for(iter in 1:N){
    # data prep
    set.seed(iter)
    train.test <- construct.train.and.test(prostate.data, response = "lpsa", split.prop = .6)
    X.train <- train.test$train %>% select(-lpsa) %>% as.matrix()
    y.train <- train.test$train %>% select(lpsa) %>% as.matrix()
    X.test <- train.test$test %>% select(-lpsa) %>% as.matrix()
    y.test <- train.test$test %>% select(lpsa) %>% as.matrix()

    # first compoute \hat{\sigma}^2 using OLS
    beta.hat.OLS <- solve(t(X.train) %*% X.train) %*% t(X.train) %*% y.train
    y.hat.OLS <- X.train %*% beta.hat.OLS
    sigma2.hat.OLS <- 1 / (nrow(X.train) - ncol(X.train)) * sum((y.train - y.hat.OLS)^2)

    # SVD of X.train
    svd.X.train <- svd(X.train)
```

```
    U <- svd.X.train$u
    V <- svd.X.train$v
    Gamma_diag <- svd.X.train$d

    # then compute the optimal lambda^*
    lambda <- ifelse(sigma2.hat.OLS >= (t(U) %*% y.train)^2, Inf, sigma2.hat.OLS * Gamma_diag^2 / ((t

    # compute the optimal ridge estimator \hat{\beta}_\lambda^* and compute held-out risk on test set
    beta.hat.ridge.optimal <- V %*% diag(Gamma_diag / (Gamma_diag^2 + lambda)) %*% t(U) %*% y.train
    held.out.risk.optimal <- 1 / nrow(X.test) * sum((y.test - X.test %*% beta.hat.ridge.optimal)^2)
    # compute the ridge estimator \hat{\beta}_\lambda^* for each \tau
    beta.hat.ridge <- matrix(NA, nrow = ncol(X.train), ncol = length(tau))
    held.out.risk <- rep(NA, length(tau))
    for(i in 1:length(tau)){
        beta.hat.ridge[,i] <- solve(t(X.train) %*% X.train + tau[i] * diag(ncol(X.train))) %*% t(X.tr
        held.out.risk[i] <- 1 / nrow(X.test) * sum((y.test - X.test %*% beta.hat.ridge[,i])^2)
    }
    # compute the risk gap
    risk.gap <- held.out.risk - held.out.risk.optimal
    risk.gap.df[, iter] <- risk.gap
}
risk.gap.df <- data.frame(t(risk.gap.df))
names(risk.gap.df) <- -10:20

# make a plot of the risk gap as a function of \tau, sort the \tau in increasing order

plotdf <- risk.gap.df %>% gather(key = "tau", value = "risk.gap") %>% mutate(tau = as.numeric(tau)) %
# boxplot for distribution and line-point plot for mean-value
plotdf %>% ggplot(aes(x = as.factor(tau), y = risk.gap)) + geom_boxplot() + geom_point(stat = "summar
```

```
## No summary function supplied, defaulting to `mean_se()`
```



We can see that that:

- for large τ value, the risk gap $\hat{r}_\tau - \hat{r}_*$ is significantly increasing and being larger than 0. Which means that in that case the 'optimal' ridge estimator $\hat{\beta}_{\lambda^*}$ out perform the regular ridge regression method (in the sense to minimize expected risk).
- But for small τ, such improvement is not significant, and the risk gap could even be negative, which means that the 'optimal' ridge estimator $\hat{\beta}_{\lambda^*}$ is slightly worse than the regular ridge regression method.