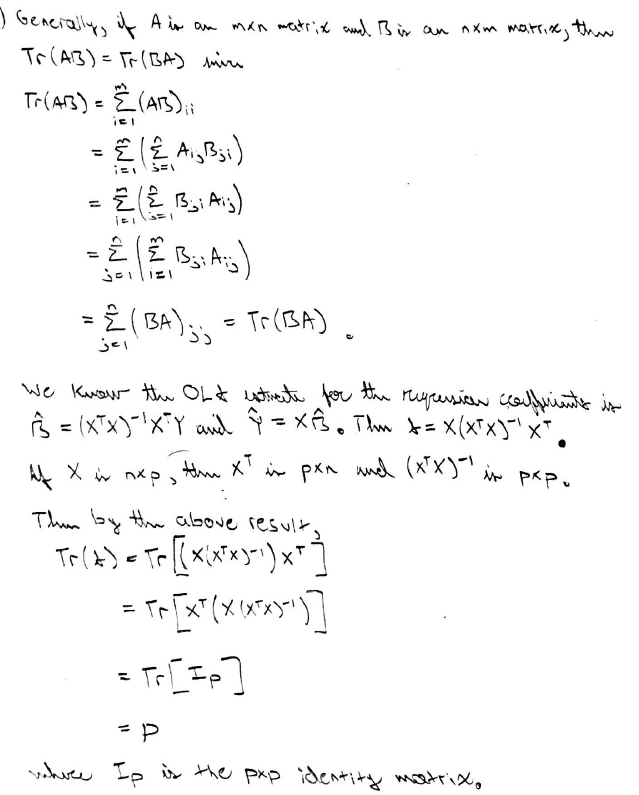
Nicholas Orriols

Statistical Learning

Homework 5

1. 
2. a) As , the integral of the square of must decrease in order to minimize the argument. If is not identically 0 then the integral term is positive, and so multiplying by makes the argument positively unbounded (so not minimized) as . Hence, we require that .

b) Similar to part a), minimizing the argument requires that the derivative of be equivalent to the zero function, which implies must be constant. In this case the integral term is (minimized to) 0, and to minimize the RSS term, we choose , which is equivalent to the intercept-only least squares model.

c) Similar to parts a) and b), minimizing the argument requires that the second derivative of be equivalent to the zero function, which implies must be linear. In this case the integral term is (minimized to) 0, and to minimize the RSS term we choose to be equivalent to the linear least squares model (since we know the linear LS model minimizes the RSS over linear functions).

d) If , then there is no smoothness constraint for minimizing the argument. Then minimizing the argument is equivalent to choosing such that RSS = 0; in this case, for each , and will likely not be very smooth.

1. a) The following code fits a regression tree, uses CV to determine the tree size, prunes the tree according to the size chosen by CV, and prints the pruned tree’s testing MSE and the following graph.

> library(tree)

> library(randomForest)

>

> train = read.csv("Problem3train.csv", header=T)

> test = read.csv("Problem3testing.csv", header=T)

> set.seed(exp(-pi / 2))

>

> fit = tree(y~., data=train)

> cv.fit = cv.tree(fit)

> cv.size = cv.fit$size[which.min(cv.fit$dev)]\*1

> prune.fit = prune.tree(fit, best=cv.size)

>

> plot(prune.fit)

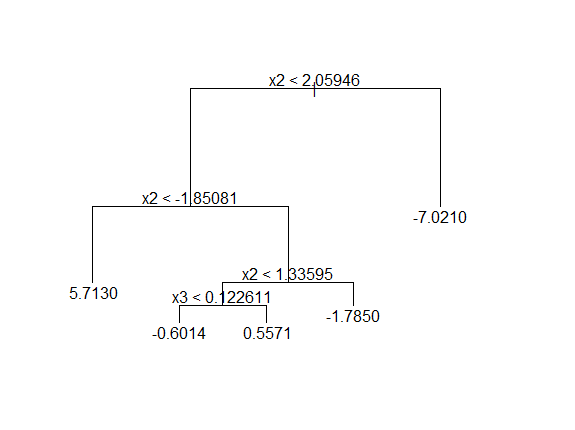
> text(prune.fit)

>

> fit.pred = predict(prune.fit, newdata = test)

> mean((fit.pred - test[,1])^2)

[1] 3.74206



b) The following code uses bagging with 1000 trees to fit a find a final regression tree. The final regression tree’s testing MSE is printed.

> fit.bag = randomForest(y~., data=train, mtry=10, ntree=1000)

> mean((predict(fit.bag, newdata=test)-test[,1])^2)

[1] 3.089651

c) The following code fits a GAM as described and outputs the testing MSE of the GAM.

> gam.fit=gam(y~s(x1,k=5,fx=T)+s(x2, k=5,fx=T)+

+ s(x3,k=5,fx=T)+s(x4, k=5,fx=T)+

+ s(x5,k=5,fx=T)+s(x6, k=5,fx=T)+

+ s(x7,k=5,fx=T)+s(x8, k=5,fx=T)+

+ s(x9,k=5,fx=T)+s(x10,k=5,fx=T),

+ data=train)

>

> gam.pred = predict(gam.fit, newdata=test)

> mean((gam.pred - test[,1])^2)

[1] 2.896434