Series 4

1. In this exercise, we will run an extensive simulation to analyze estimates of the test MSE of several cross-validation type methods. Assume we have n independent samples of the random vector of (X, Y) where $X \in \mathbb{R}^p$ and $Y \in \mathbb{R}$ are random variables. We assume that

$$Y = f(X) + \varepsilon, \tag{1}$$

where ε is a random noise variable that is independent of X, and $f : \mathbb{R}^p \to \mathbb{R}$ is a deterministic function. In the lecture, we have seen different regression methods that yield an estimate $\widehat{f}$ of the function f based on the observed samples, such as linear regression, polynomial regression, K-nearest-neighbour regression and LOESS.

a) We assume that $X = (X_1, \cdots, X_p)$, where $X_i = g(Z_i)$, $i \in \{1, \cdots, p\}$, and $Z_1, \cdots, Z_p$ are independent uniform random variables on $[-1, 1]$ and $g : [-1, 1] \to [-1, 1]$ is a deterministic function. Where not differently stated, we will take $p = 2$. Simulate $n = 500$ samples of X (for $p = 2$) and each of the following functions for g.

```
> g1<-function(x){2*x/(1+abs(x)^1.5)} # Favour x-values with larger absolute value
> g2<-function(x){x^3/abs(x)^1.5}     # Favour x-values with smaller absolute value
> g3<-function(x){x}                  # Keep the uniform distribution
```

Save the samples in a $n \times p$ matrix `X` where each row represents a sample. Then plot the two-dimensional samples for $g =$ `g1`, $g =$ `g2` and $g =$ `g3`.

R-hint:

```
> x<-runif(n*p,min=-1,max=1)
> Z<-matrix(x,ncol=p)
> X<-g1(Z)
```

b) Now choose one of `g1`, `g2` or `g3`, and write a function `sampleX()` that generates a matrix of $n = 500$ samples `X` as done in part a).

R-hint: `sampleX<-function(n=500){...`

c) We assume that f has the form $f : x \mapsto$ `f1dim`(x_1), where `f1dim()` is the R-function

```
> f1dim<-function(x){ sin(8*x)/(1+(4*x)^2) }
```

Write an R-function `f` that takes an $n \times 2$ matrix `X` as returned by `sampleX()`, and maps it to a vector $(f($`X[1,]`$), \cdots, f($`X[n,]`$))^\intercal$, i.e., the vector of function values of the samples in `X`.

d) We assume that the noise variable ε is normally distributed with mean 0 and standard deviation σ. Choose a $\sigma \in (0, 1]$. Then write an R-function `sampleY(X)` that takes an $n \times 2$ matrix `X` of samples of X as returned by `sampleX()`, and samples a vector of corresponding values of Y according to model (1). Use the function `f` from part c).

R-hint: `sampleY<-function(X){ return(f(X)+rnorm(???,???))}`

e) We first choose k-nearest neighbour as the regression method with $k = 8$. We install and load the package `kknn`, and illustrate its usage.

```
> Xtrain<-sampleX()                       # some training data
> Ytrain<-sampleY(Xtrain)
> dfTrain=data.frame(y=Ytrain,x=Xtrain) # wrap in a dataframe
> Xtest<-sampleX()                        # same for test data
> Ytest<-sampleY(Xtest)
> dfTest=data.frame(x=Xtest)
> fit.kknn <- kknn(y ~ ., dfTrain,dfTest,k=8)
> predTest=predict(fit.kknn)              # predictions on dfTest
```

Use simulation to approximate the true expected test MSE of the 8 nearest neighbour regression method.
Hint: Simulate $M = 1000$ times training (500 samples) and test data (2000 samples), fit the estimator using the training data and evaluate it on the test data. Average the M obtained test MSEs.

f) Consider the following methods to estimate the expected test MSE of the chosen regression method (KNN with $k = 8$):

- Validation set approach
- Repeated Validation set approach (take the average of 10 estimates of the validation set approach, each time with random a random partition of the samples, one half for training and the other half for estimation of the test MSE)
- 10-Fold Cross Validation
- Repeated 10-Fold Cross Validation (take the average of 10 estimates of 10-Fold CV, each time with a random partition of the samples into 10 folds)
- Leave-one-out Cross Validation

For each method, write an R-function that computes the corresponding estimate of the test MSE taking arguments `X` and `Y` as returned by the functions `sampleX()` and `sampleY(X)`.
R-hint: The first function should look something like this:

```
> ValidationSet<-function(X,Y){
        n<-length(Y)
        s <- sample(1:n, size=n, replace=F)
        folds <- cut(seq(1,n), breaks=2, labels=FALSE)
        ind.test <- s[which(folds==1)]
        dfTrain=data.frame(y=Y[ind.test],x=X[ind.test,])
        dfTest=data.frame(x=X[-ind.test,])
        fit.kknn <- kknn(y ~ ., dfTrain,dfTest,k=8)
        predTest=predict(fit.kknn)
        Ytest<-Y[-ind.test]
        MSEEstimate=mean((predTest-Ytest)^2)
        return(MSEEstimate)
 }
> # usage
> X <- sampleX()
> Y <- sampleY(X)
> ValidationSet(X,Y)
```

The function for the repeated validation set approach can then be based on this.

```
> RepeatedValidationSet<-function(X,Y){
        MSEEstimate <- replicate(10, ValidationSet(X,Y))
        return(mean(MSEEstimate))
 }
```

g) Use simulation to approximate the distribution of the estimators for the expected test MSE from the previous subtask. Use a boxplot to visualize your results. Also add a horizontal line corresponding to the approximated true expected test MSE from part e).
R-hint: Understand and use the following function which returns the estimates of the expected test MSE for a specified estimation function on randomly sampled `X` and `Y`.

```
> EvaluateOnSimulation<-function(estimationFunction, iterations=200){
     result<-numeric(iterations)
     for (i in 1:iterations) {
         X<-sampleX()
         Y<-sampleY(X)
         result[i]= estimationFunction(X,Y)
     }
     return(result)
 }
> EstimatesVS <- EvaluateOnSimulation(ValidationSet) #use like this
```

For the boxplot, use your estimates as follows:

```
> Estimates <- cbind(EstimatesVS,...) #results from the 5 CV methods
> boxplot(Estimates)
```

To add a horizontal line, use `abline(h=...)`.

h) Use the results of the previous subtask to approximate the bias and variance of the corresponding estimators.
Hint: To approximate the bias, you need your result from part e).

i) Modify your 10-Fold CV function to output an estimate of the variance of the estimated expected test MSE using the formula

$$\frac{1}{10}\mathrm{Var}(\mathrm{MSE}_1, \cdots, \mathrm{MSE}_{10}),$$

where Var() is the sample variance and MSE_i is the estimated test MSE on the i-th fold, $i \in \{1, \dots 10\}$. This estimator is often used in practice.

j) There is some dispute in the literature whether LOOCV or 10-Fold CV has a larger variance. We wish to collect your results for a variety of different settings. Change one or more of the following:

- The number of predictors p
- The distribution of the predictors, i.e. choose another $g \in \{\mathtt{g1}, \mathtt{g2}, \mathtt{g3}\}$ or something else
- The function f
- The noise distribution
- The regression method, i.e. use for example linear regression, LOESS, polynomial regression instead of KNN or simply change the parameter k for KNN

Then repeat the analysis of tasks e) and h) for 10-Fold CV and LOOCV, and input the results in a Google form :`https://goo.gl/forms/6MrI8Wpx7dR3SJmC3`. We can then see how the variance of LOOCV and 10-Fold CV compares.

Preliminary discussion: Friday, March 22.

Deadline: Friday, March 29.