Series 10

1. For a dataset (x_i, y_i) $(i = 1, \ldots, n)$ with n observations and predictors $x_i \in \mathbb{R}^p$, the leave-one-out cross validation (LOOCV) mean-squared error is:

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i^{(-i)})^2,$$

where $\hat{y}_i^{(-i)}$ denotes the fitted value in x_i of the model trained on the whole dataset but the i^{th} observation, that is, on dataset (x_j, y_j) $(j = 1, \ldots, i-1, i+1, \ldots, n)$. Computing the leave-one-out cross validation error can be computationally intensive due to fitting n different models. In some specific cases, it happens that a closed formula for LOOCV mean-squared error exists requiring no additional fit than the global one (on the whole dataset).

a) Consider the general case of a linear fitting operator S such that

$$Sy = (\hat{m}(x_1), \ldots, \hat{m}(x_n))^T,$$

where $y = (y_1, \ldots, y_n)^T$ and S is a matrix of size $n \times n$.
We assume that we can obtain the prediction $\hat{m}^{(-i)}(x_i)$ on the i^{th} observation of the smoother fitted on the whole dataset but the i^{th} observation by the formula

$$\hat{m}^{(-i)}(x_i) = \frac{(Sy)_i - S_{ii}y_i}{1 - S_{ii}}. \tag{1}$$

Show that the following formula holds

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{m}^{(-i)}(x_i))^2 = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{m}(x_i)}{1 - S_{ii}}\right)^2.$$

b) Linear regression is a first sub-case where we can get a formula for the LOOCV mean-squared error. We denote the *hat* $H \in \mathbb{R}^{n\times n}$ matrix by

$$H = X(X^TX)^{-1}X^T,$$

where X is the $n \times p$ design matrix. Show that the formula in Equation (1) is valid for linear regression.
Hints:

- consider the dataset where we replace the i^{th} observation (x_i, y_i) by the observation $(x_i, \hat{y}_i^{(-i)})$ with $\hat{y}_i^{(-i)}$ being the fitted value at x_i from the model trained without the i^{th} observation (x_i, y_i).
- justify why $\hat{y}_i^{(-i)}$ coincides with $\hat{z}_i$ where $\hat{z} = Hz$ and z is a vector of responses defined as $z = (z_1, \ldots, z_p)$ with $z_j = y_j$ if $j \neq i$ and $z_i = \hat{y}_i^{(-i)}$ (the two fitted values coincide).
- deduce Equation (1) for linear regression from the last observation.

2. In this exercise we will get a grasp of the curse of dimensionality. Let $X \in \mathbb{R}^n$ be uniformly distributed within the unit ball in $\mathbb{R}^n$ (with respect to the L^2 norm). Suppose we have n independent and identically (i.i.d.) copies of X: $X_1, \ldots, X_n$. Let $X_{(1)}$ denote the one that is closest to the origin (again in terms of the L^2 norm), and let $\|X_{(1)}\|_2$ be its L^2 distance to the origin.

a) Determine $P(\|X_{(1)}\|_2 > t)$ as a function of n and t, where $0 \leq t \leq 1$.
Hints:

- thinking first in one dimension could help;

- the volume in $\mathbb{R}^n$ of an L^2 ball with radius R is given by

$$V_n(R) = C(n)R^n,$$

where $C(n)$ is a constant depending on n.

b) Determine the median of $\|X_{(1)}\|_2$ as a function of n.

c) Compute the median of $\|X_{(1)}\|_2$ for $n = 500$. What does this tell you about the distribution of points within the unit ball?

3. In this exercise, we want to compare the impact of the curse of dimensionality on the expected test error for three different models: k-nearest neighbors (KNN), multiple linear regression and generalized additive models with splines (GAM). We consider the specific case when the underlying signal is sparse (only one predictor has an effect on the response) and the signal is non-linear.

a) Construct training and test sets in which the response is non-linear in the first predictor with the help of the following code. Please also install or load the R libraries below that will be needed for the exercise:

```
> # libraries needed
> # for GAMs
> library(gam)
> # for KNN regression
> library(FNN)
> # for repro
> set.seed(1)
> # predictors in training set
> xtrain <- matrix(rnorm(20*100),ncol=20, nrow = 100)
> # response in training set
> ytrain <- sin(2*xtrain[,1]) + 0.3*rnorm(100)
> # training set
> dtrain <- data.frame(xtrain,y = ytrain)
> # predictors in test set
> xtest <- matrix(rnorm(20*100),ncol=20, nrow = 100)
> # response in test set
> ytest <- sin(2*xtest[,1]) + 0.3*rnorm(100)
> # test set
> dtest <- data.frame(xtest,y = ytest)
```

b) Fit on the training set each of the three models (KNN, multiple linear regression and GAM with splines) with only the first predictor. For the GAM you will use a spline with degree of freedom 4 (function `s(..., 4)` in the formula) and for the KNN regression you will vary the parameter k (number of nearest neighbors) between 1 and 10 (fit 10 KNN). Record the test mean-squared error for each approach and discuss your results.

c) Apply again the same procedure as in b), but now add incrementally additional predictors in the fit that are not associated with the response (columns 2 to 20 in the dataset). Record the test mean-squared error for each possible number m of additional predictor between 1 (one additional predictors) and 19 (all predictors used in the fit), for each of the 12 modelling approaches (GAM with splines of degree 4 for each predictor, multiple linear regression and 10 KNN fits with varying parameter k between 1 and 10).

d) Plot the test mean-squared error curves as a function of $1/k$ (for GAM and multiple linear regression just plot a straight line) and discuss the behaviour of the test error when m varies between 1 and 19.

4. The backfitting algorithm is a simple iterative procedure used to fit generalized additive models. In this exercise we will get a grasp of this algorithm by applying it to multiple linear regression (which is a specific sub-case of GAM). Let us consider multiple linear regression with p covariates

$$y = \beta_0 + \sum_{i=j}^{p} \beta_j x_j + \epsilon.$$

In this case the algorithm boils down to:

1. compute the overall mean $\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} y_i$.
2. initialize vectors $\hat{g}_1, \ldots, \hat{g}_p \in \mathbb{R}^n$ with 0 values.
3. until convergence, iteratively cycle over $k = 1, \ldots, p$ and fit $\hat{\beta}_k$ the model

$$y - \hat{\mu} - \sum_{j:j\neq k} \hat{g}_j = \alpha_k + \beta_k x_k + \epsilon,$$

   and update $\hat{g}_k$ as

$$\hat{g}_k = \hat{\alpha}_k + \hat{\beta}_k x_k.$$

4. output $\hat{\beta}_0, \ldots, \hat{\beta}_p$ as estimates of the regression coefficients, where $\hat{\beta}_0$ is the combined estimate of the intercept: $\hat{\beta}_0 = \hat{\mu} + \hat{\alpha}_1 + \ldots + \hat{\alpha}_p$.

a) Implement the backfitting algorithm for the following two synthetic datasets (given in code below) and compare the resulting estimated coefficients with the multiple linear regression estimates. Try different orders when you cycle through the variables also.

```
> # for reproducibility
> set.seed(1)
> n <- 100
> # first dataset
> x1 <- rnorm(n)
> x2 <- rnorm(n)
> x3 <- rnorm(n)
> x4 <- rnorm(n)
> y <- 0.7 + x1 + 2*x2 + 0.5*x3 -3*x4 + rnorm(n)
> # second dataset
> x1_cor <- rnorm(n)
> x2_cor <- rnorm(n)
> x3_cor <- rnorm(n)
> x4_cor <- x1_cor + 0.4*rnorm(n)
> y_cor <- x1_cor + 2*x2_cor + 0.5*x3_cor -3*x4_cor + rnorm(n)
```

Preliminary discussion: Friday, May 17.

Deadline: Friday, May 24.