Series 5

1. Consider the `diabetes`-dataset from the lecture notes (Section 3.2) and the model

$$Y_i = m(X_i) + \epsilon_i,$$

where the response Y is a log-concentration of a serum (in connection with diabetes) and the predictor variable X is the age in months of children.

We want to know if a complicated nonparametric regression gives us valuable information and which one is the best. The generalization error of the following fits should be compared:

1. the kernel regression fit from `ksmooth`,
2. the local polynomial fit from `loess`,
3. a smoothing spline fit from `smooth.spline` with a fixed degree of freedom,
4. a smoothing spline fit from `smooth.spline` with the smoothing parameter selected automatically by cross-validation,
5. a constant "fit" by the overall mean of Y_i, simply ignoring the X_i-values.

a) Read in the dataset as follows:

```
> diabetes <-
   read.table("http://stat.ethz.ch/Teaching/Datasets/diabetes2.dat",
              header = TRUE)
> reg <- diabetes[, c("Age", "C.Peptide")]
> names(reg) <-     c("x",   "y")
```

Sort the observations along x, for easier dealing with the hat matrix:

```
> reg <- reg[sort.list(reg$x), ]
```

Plot the data; choose a reasonable bandwidth h for a Nadaraya-Watson kernel estimator by eye. Perform a non-parametric regression on the dataset using the kernel estimator of `ksmooth` with the bandwidth h of your choice.
Calculate the leave-one-out CV-score of your fit; the CV-score is an estimator for the generalization error. Because you will have to do that for the other regression methods too, it is recommended to write a utility function for calculating the CV-score. An R-skeleton for such a function is available on the course homepage, but you can of course also implement everything on your own from scratch.
Finally, calculate the degrees of freedom `df.nw` that correspond to your fit. For that aim, calculate the hat matrix and its trace, cf. Formula (3.6) of the lecture notes and the above-mentioned code skeleton (you can also use `hatMat` from the package `sfsmisc`) We will use the same degrees of freedom as smoothing parameter for the other regression methods.

b) Perform a non-parametric regression on the dataset `diabetes` using the local polynomial fit from `loess`. Use the degrees of freedom `df.nw` from task a) as a smoothing parameter:
`loess(..., enp.target = df.nw, surface = "direct")`
The last argument, `surface = "direct"`, ensures that you can also predict values outside the range of x-values you used for estimation.
Calculate the CV-value for this estimator, too.

c) Perform a non-parametric regression on the dataset `diabetes` using a smoothing splines fit from `smooth.spline`; use the degrees of freedom `df.nw` from task a).
`smooth.spline` can calculate the CV-value internally; start by looking at that value:

```
> est.ss <- smooth.spline(..., cv = TRUE, df = df.nw)
> est.ss$cv.crit
```

Compare this internally calculated value with a value calculated "on your own", i.e. similarly to the CV-values in tasks a) and b). When doing your own calculations, do not provide the smoothing parameter via the argument `df`, but use the parameter `spar` from the automatic calculation above as follows:
`smooth.spline(..., spar = est.ss$spar)`
Defining `spar` instead of `df` will make the computation faster. You may also take advantage of Formula (4.5) in the lecture notes to calculate the CV-value. (**Hint**: we expect that the calculation using `loocv` slightly differs from the automated calculation and the one using Formula (4.5).)

d) Perform a non-parametric regression on the dataset `diabetes` using a smoothing splines fit from `smooth.spline`, this time using automatically selected degrees of freedom. Report the CV-value that is calculated internally:
`> smooth.spline(..., ..., cv = TRUE)$cv.crit`

e) Perform a constant fit of the data, i.e. neglect the x-values and calculate the mean of the y-values. Calculate the corresponding CV-value also for that estimator.

f) The CV-scores calculated in tasks a) to e) are estimators of the generalization error. According to the values you calculated, which of the five estimators has the lowest generalization error? The comparison of the CV-value of method no. 4 (smoothing splines with automatically selected degrees of freedom) with the others is not fair. Can you explain the problem?

2. In this excercise, we construct different bootstrap confidence intervals and check their empirical coverage for different sample sizes in a simulation study. An R-skeleton including hints is available on the course website.

a) We want to estimate the trimmed mean θ of the Gamma distribution where the 10% largest and 10% smallest observations are trimmed. Approximate θ based on a very large sample. (Hint: See the R-skeleton to find the values for "shape" and "rate" of the Gamma distribution that we will use.)

b) Now construct a sample of size 40 from the given Gamma distribution and estimate θ using the sample trimmed mean $\hat{\theta}$.

c) Construct three different 95%-bootstrap confidence intervals (CI) for the trimmed mean θ based on the sample from task b). The three CIs are:

- "quantile": $\left[q_{\hat{\theta}^*}(\alpha/2),\ \ q_{\hat{\theta}^*}(1-\alpha/2)\right]$,
- "normal approximation": $2\hat{\theta} - \overline{\hat{\theta}^*} \pm q_Z(1-\alpha/2)\cdot \widehat{sd}(\hat{\theta})$ where $Z \sim \mathcal{N}(0,1)$,
  $\widehat{sd}(\hat{\theta}) = \sqrt{\frac{1}{R-1}\sum_{i=1}^{R}\left(\hat{\theta}^{*i} - \overline{\hat{\theta}^*}\right)^2}$
  Remark: One may expect the "intuitive" formula $\hat{\theta} \pm q_Z(1-\alpha/2)\cdot\widehat{sd}(\hat{\theta})$. However, a bias correction is applied in this intuitive formula. The estimated bias is given by $\overline{\hat{\theta}^*} - \hat{\theta}$. If we subtract this estimated bias in the intuitive CI formula, we obtain $\hat{\theta} - (\overline{\hat{\theta}^*} - \hat{\theta}) \pm q_Z(1-\alpha/2)\cdot \widehat{sd}(\hat{\theta})$, which coincides with the "normal approximation" CI given above.
- "reversed quantile": $\left[\hat{\theta} - q_{\hat{\theta}^*-\hat{\theta}}(1-\alpha/2),\ \ \hat{\theta} - q_{\hat{\theta}^*-\hat{\theta}}(\alpha/2)\right]$, and

Hint: See R-skeleton. If you want to use the R package `boot`, `type = "perc"` corresponds to "quantile" bootstrap CI, `type = "norm"` corresponds to "normal approximation" bootstrap CI, and `type = "basic"` corresponds to "reversed quantile" bootstrap CI.

d) To investigate the performance of the different confidence intervals, we conduct a small simulation study. Simulate 200[1] new data sets (40 observations each) and construct the different bootstrap CIs based on 500 bootstrap replicates. For each type of CI, compute the percentage of CIs that do not contain θ. Specifically, if the CI is denoted by $(CI_l,\ CI_u)$, compute the percentage of times that $\theta < CI_l$ and the percentage of times that $\theta > CI_u$ (non-coverage rate of the upper and lower end of the CI). Ideally, both percentages should be 2.5%.

e) Repeat task d) for sample sizes $n = 10, 40, 160, 640$ and plot the upper and lower (one-sided) non-coverage as a function of n in two separate plots. See task d) for the definition of the non-coverage. What do you observe?

Preliminary discussion: Friday, April 01.

Deadline: Friday, April 08.

[1]Start with a smaller number of data sets and / or number of bootstrap replicates to try your code and to see if your code is running correctly.