Problem Sets for Nonparametric Statistics

MSc in Statistics for Data Science at Carlos III University of Madrid

Eduardo García-Portugués

2021-03-13, v1.0.4

Instructions

Do the three exercises your group has been assigned to do in the table below. Each one corresponds to a different category (A, B, or C). The `NIU_submitter` is the student's ID of whomever sent the Google form indicating the group composition. Use the `Group number` to name your files, as indicated in the submission instructions. Read the exercises carefully and, if required, look for their context in the notes. In case any group member has previously delivered as a voluntary exercise any of the group's assigned exercises, please communicate in order to reassign the conflicting exercises.

Group number	NIU_submitter	A	B	C
1	100444499	6	4	4
2	100441742	5	5	9
3	100444982	6	5	3
4	100345869	9	8	2
5	100445061	1	3	2
6	100425253	1	7	8
7	100414573	9	2	3
8	100442024	2	4	6
9	100354390	1	2	1
10	100447036	4	7	7
11	100441618	3	3	9
12	100446255	2	6	4
13	100435745	4	8	7
14	100446954	8	7	5
15	100279487	3	1	5
16	100445616	6	3	1
17	100439245	7	6	4
18	100443596	7	2	8
19	100448075	8	6	6
20	100423743	5	5	1

Exercises

Category A (3.0 points)

1. **Exercise 2.25.** Consider the `MNIST` dataset in the `MNIST-tSNE.RData` file. It contains the first $60,000$ images of the MNIST database. Each observation is a grayscale image made of 28×28 pixels that is recorded as a vector of length $28^2 = 784$ by concatenating the pixel columns. The entries of these vectors are numbers in $[0, 1]$ indicating the level of grayness of the pixel: 0 for white, 1 for black. These

vectorised images are stored in the $60,000 \times 784$ matrix in `MNIST$x`. The 0–9 labels of the digits are stored in `MNIST$labels`.

a. Compute the average gray level, `av_gray_one`, for each image of the digit "1".
b. Compute and plot the kde of `av_gray_one`, taking into account that it is a positive variable.
c. Overlay the lognormal distribution density, with parameters estimated by maximum likelihood (use `MASS::fitdistr`).
d. Repeat c for the Weibull density.
e. Which parametric model seems more adequate?

2. **Exercise 3.10.** The `data(sunspots_births, package = "rotasym")` dataset contains the recorded sunspots births during 1872–2018 from the Debrecen Photoheliographic Data (DPD) catalogue. The dataset presents 51,303 sunspot records, featuring their positions in spherical coordinates (`theta` and `phi`), sizes (`total_area`), and distances to the center of the solar disk (`dist_sun_disc`).

   a. Compute and plot the kde for `phi` using the DPI selector. Describe the result.
   b. Compute and plot the kernel density derivative estimator for `phi` using the adequate DPI selector. Determine approximately the location of the main mode(s).
   c. Compute the log-transformed kde with `adj.positive = 1` for `total_area` using the NS selector.
   d. Draw the histogram of $M = 10,000$ samples simulated from the kde obtained in a.

3. **Exercise 4.20.** Consider the `data(sunspots_births, package = "rotasym")` dataset. Then:

   a. Filter the dataset to account only for the 23rd `cycle`.
   b. Inspect the graphical relation between `dist_sun_disc` (response) and `log10(total_area)` (predictor).
   c. Compute the CV bandwidth for the above regression, for the local constant estimator.
   d. Compute and plot the local constant estimator using CV bandwidth. Comment on the fit.
   e. Repeat c and d for the local linear estimator.

4. **Exercise 5.7.** The `data(ChickWeight)` dataset in R contains 578 observations of the `weight`, `Time`, and `Diet` of chicks.

   a. Consider the regression `weight ~ Time + Diet`. Fit the local linear estimator with CV bandwidths by taking the nature of the variables into account.
   b. Plot the marginal effects of `Time` for the four possible types of `Diet`. Overlay the four samples and using different colors.
   c. Obtain the bootstrap 90%-confidence intervals of the expected weight of a chick on the 20th day, for each of the four types of diets (take $B = 500$). With a 90% confidence, state which diet dominates in terms of weight the other diets, and which are comparable.
   d. Produce a plot that shows the extrapolation of the expected weight, for each of the four diets, from the 20th to the 30th day. Add bootstrap 90%-confidence bands.

5. **Exercise 5.10.** The dataset at http://www.stat.cmu.edu/~larry/all-of-nonpar/=data/bpd.dat (alternative link) contains information about the presence of bronchopulmonary dysplasia (binary response) and the birth weight in grams (predictor) of 223 newborns.

   a. Use the three approaches described above to compute the local logistic regression (first degree) and plot their outputs for a bandwidth that is somehow adequate.
   b. Using $\hat{h}_{\mathrm{LCV}}$, explore and comment on the resulting estimates, providing insights into the data.
   c. From the obtained fit, derive several simple diagnostic rules for the probability of the presence of bronchopulmonary dysplasia from the birth weight.

6. **Exercise 5.11.** The `challenger.txt` dataset contains information regarding the state of the solid rocket boosters after launch for 23 shuttle flights prior the Challenger launch. Each row has, among others, the variables `fail.field` (indicator of whether there was an incident with the O-rings), `nfail.field` (number of incidents with the O-rings), and `temp` (temperature in the day of launch, measured in Celsius degrees).

a. Fit a local logistic regression (first degree) for `fails.field ~ temp`, for three choices of bandwidths: one that oversmooths, another that is somehow adequate, and another that undersmooths. Do the effects of `temp` on `fails.field` seem to be significant?
b. Obtain $\hat{h}_{\text{LCV}}$ and plot the LCV function with a reasonable accuracy.
c. Using $\hat{h}_{\text{LCV}}$, predict the probability of an incident at temperatures -0.6 (launch temperature of the Challenger) and 11.67 (specific recommendation by the vice president of engineers).
d. What are the local odds at -0.6 and 11.67? Show the local logistic models about these points, in spirit of Figure 5.1, and interpret the results.

7. **Exercise S-I.** The cellphone company S has recently suffered a production failure in the battery of one of their flagship smartphones, code-named S-7. The failure drives the temperature of the battery to increase excessively, up to the point of a potential combustion. S claims that "only 0.1% of the S-7s are susceptible of presenting anomalies in the battery". This statistic is based on the cellphones that have been reported as defective because they have *already* burned, therefore is not taking into account the smartphones with defective battery at risk of future combustion. The rival company, A, receives internal information leaked by an anonymous source in S. The information consists on two samples of the working temperatures of past smartphones (`temps-other.txt`) and of the problematic smartphone (`temps-7.txt`). As the lead data scientist of A, you are tasked to dig into these two samples to evaluate the claim of S.

   a. Perform a kernel density estimation for `temps-7` and `temps-other` using what you consider is the most adequate bandwidth. Since the temperatures are positive, is it required to perform any transformation?
   b. Is there any important difference on the results from considering the LSCV selector over the DPI selector?
   c. It seems that in `temps-7` there is a secondary mode. Compute a kernel derivative estimation for `temps-7` and `temps-other` using what you consider are the most adequate bandwidths.
   d. Precisely determine the location of the extreme points.
   e. Check with a kernel second derivative that the extreme points are actually modes.

8. **Exercise S-II.** Consider the context given in Exercise S-I.

   a. Obtain the estimated smallest set that contains the 99% of the probability for `temps-7` and `temps-other`. Use what you consider is the most adequate bandwidth. What are your conclusions?
   b. Tweaking the percentage of probability in a, what is your most precise conclusion about the percentage of defective smartphones?

   To perform, a–b, construct a function `kde_level_set2` that adapts the function `kde_level_set` to:

   - Receive as main arguments a sample `x` and a bandwidth `h`, instead of a `density` object.
   - Use internally `ks::kde` instead of `density`.
   - Receive an α, instead of c, and internally determine $\hat{c}_\alpha$.
   - Retain the functionality of `kde_level_set` and return also `c_alpha_hat`.

9. **Exercise 5.5.** Implement an R function to compute (5.9) for the local constant estimator with multivariate (continuous) predictor. The function must receive as arguments the sample $(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)$ and the bandwidth vector $\mathbf{h}$. Use the normal kernel. Test your implementation by:

   1. Simulating a random sample from a regression model with two predictors.
   2. Computing its cross-validation bandwidths via `np::npregbw`.
   3. Plotting a contour of the function $(h_1, h_2) \mapsto \text{CV}(h_1, h_2)$ and checking that the minimum of this surface and the solution given by `np::npregbw` coincide.

Category B (3.0 points)

1. **Exercise 2.18.** Implement the h_{MISE} using (2.23) and (2.33) for model `nor1mix::MW.nm5`. Then, compute by Monte Carlo the densities of $\hat{h}_{\text{DPI}}/h_{\text{MISE}} - 1$, $\hat{h}_{\text{LSCV}}/h_{\text{MISE}} - 1$, and $\hat{h}_{\text{BCV}}/h_{\text{MISE}} - 1$.

Compare them for $n = 100, 200, 500$, adding a vertical line to represent the h_{MISE} bandwidth. Describe in detail the results and the major takeaways.

2. **Exercise 2.19.** Compare the MISE and AMISE criteria in three densities in `nor1mix` of your choice. To that purpose, code (2.33) and the AMISE expression for the normal kernel, and compare the two error curves. Compare them for $n = 100, 200, 500$, adding a vertical line to represent the h_{MISE} and h_{AMISE} bandwidths. Describe in detail the results and the major takeaways.

3. **Exercise 4.3.** Implement in R your own version of the Nadaraya–Watson estimator and compare it with `nw`. Focus only on the normal kernel and reduce the accuracy of the final computation up to `1e-7` to achieve better efficiency. Are you able to improve the speed of `nw`? Use the `microbenchmark::microbenchmark` function to measure the running times for a sample of size $n = 10,000$.

4. **Exercise 4.9.** Perform the following tasks:

   a. Code your own implementation of the local cubic estimator. The function must take as input the vector of evaluation points `x`, the sample `data`, and the bandwidth `h`. Use the normal kernel. The result must be a vector of the same length as `x` containing the estimator evaluated at `x`.
   b. Test the implementation by estimating the regression function in the location model $Y = m(X) + \varepsilon$, where $m(x) = (x-1)^2$, $X \sim \mathcal{N}(1, 1)$, and $\varepsilon \sim \mathcal{N}(0, 0.5)$. Do it for a sample of size $n = 500$.

5. **Exercise 4.18.** Perform the following simulation study:

   1. Simulate $n = 100$ observations from the regression model $Y = m(X) + \varepsilon$, where X is distributed according to a `nor1mix::MW.nm2`, $m(x) = x\cos(x)$, and $\varepsilon + 1 \sim \text{Exp}(1)$.
   2. Compute the LSCV bandwidths for the local constant and linear estimators.
   3. Repeat steps 1–2 $M = 200$ times. *Trick*: use `options(np.messages = FALSE)` or `capture.output` omit the messages from `np::npregbw`, and `txtProgressBar` and `setTxtProgressBar` to have an indication of the progress of the simulation.

   Compare graphically the two samples of LSCV bandwidths. To do so, show in a 1×2 plot the log-transformed kdes for the two samples (set `xlim = c(0, 2)`), employ `rug` for both of them, and add vertical bars indicating where the sample medians are located. The display should look like Figure 4.5.

6. **Exercise 4.19.** Perform the following simulation study for each of the local constant and local linear estimators.

   1. Plot the regression function $m(x) = x\cos(x)$.
   2. Simulate $n = 100$ observations from the regression model $Y = m(X) + \varepsilon$, where $\frac{1}{2}X$ is distributed according to a `nor1mix::MW.nm7` and $\varepsilon \sim \mathcal{N}(0, 1)$.
   3. Compute the LSCV bandwidths and the fit associated with this bandwidth.
   4. Plot the fit as a line. *Trick*: use `rainbow` with `alpha` in order to change colors and adjust the transparency level.
   5. Repeat steps 2–4 $M = 75$ times.

   The two outputs should be similar to Figure 4.5.

7. **Exercise 4.20.** Repeat Exercise 4.18 replacing step 3 with:

   3. Compute the variable bandwidths `bwtype = "generalized_nn"` and `bwtype = "adaptive_nn"` and the fits associated with them.

   Describe the obtained results. Are they qualitatively different from those based on fixed bandwidths, given in Figure 4.5? Do the fits improve with variable bandwidths?

8. **Exercise 5.2.** Implement a function to compute $\hat{m}(\mathbf{x}; 1, h)$ for an arbitrary dimension p. The function must receive as arguments the sample $(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)$, the bandwidth vector $\mathbf{h}$, and a collection of evaluation points $\mathbf{x}$. Implement the function by computing the design matrix $\mathbf{X}$, the weight matrix $\mathbf{W}$, the vector of responses $\mathbf{Y}$, and then applying (5.6). Test the implementation in the example below.

Category C (4.0 points)

1. **Exercise 3.23.** Load the `wines.txt` dataset and perform a PCA removing the `vintage` variable and after standardizing the variables.

   a. How many clusters are detected using three PCs? What happens if `min.clust.size` is set such that the minimum cluster contains more than 5% of the sample? Does it seem like a sensible choice to set such value of `min.clust.size`?
   b. Using the same `min.clust.size`, consider six PCs. How many clusters are detected? Perform a pairs scatterplot and identify the PCs that are useful to identify clusters and the ones that seem to be adding noise.
   c. Do the clustering with two PCs. How many clusters are now detected? Compare the clustering with two and three PCs. Why do you think there are fewer clusters in the latter?
   d. Compare the cluster made in a with the variable `vintages`.
   e. Run `kmeans` on the data used in a with $k = 3, 4$ and compare the results graphically.

2. **Exercise 3.24.** Consider the `MNIST` dataset described in Exercise 2.25. Investigate the different ways of writing the digit "3" using kernel mean shift clustering. Follow the next steps:

   a. Consider the t-SNE scores `y_tsne` of the class "3". Use only the first $2,000$ t-SNE scores for the class "3" for the sake of computational expediency.
   b. Compute the normal scale bandwidth matrix that is appropriate for performing kernel mean shift clustering with the first $2,000$ t-SNE scores.
   c. Do kernel mean shift clustering on that subset using the previously obtained bandwidth and obtain the modes $\boldsymbol{\xi}_j$, $j = 1, \ldots, M$, that cluster the t-SNE scores.
   d. Determine the M images that have the closest t-SNE scores, using the Euclidean distance, to the modes $\boldsymbol{\xi}_j$.
   e. Show the closest images associated with the modes. Do they represent different forms of drawing the digit "3"?

3. **Exercise 3.29.** Consider the `MNIST` dataset described in Exercise 2.25. Classify the digit images, via the t-SNE scores `y_tsne`, into the digit labels:

   a. Split the dataset into the training sample, comprised of the first $50,000$ t-SNE scores and their associated labels, and the test sample (rest of the sample).
   b. Using the training sample, compute the plug-in bandwidth matrices for all the classes.
   c. Use these plug-in bandwidths to perform kernel discriminant analysis.
   d. Plot the contours of the kernel density estimator of each class and overlay the t-SNE scores as points. Use coherent colors between contours and points, and add a legend.
   e. Obtain the successful classification rate of the kernel discriminant analysis.

4. **Exercise 3.30.** Load the `ovals.RData` file.

   a. Split the dataset into the training sample, comprised of the first $2,000$ observations, and the test sample (rest of the sample). Plot the dataset with colors for its classes. What can you say about the classification problem?
   b. Using the training sample, compute the plug-in bandwidth matrices for all the classes.
   c. Use these plug-in bandwidths to perform kernel discriminant analysis.
   d. Plot the contours of the kernel density estimator of each class and the classes partitions. Use coherent colors between contours and points.
   e. Predict the class for the test sample and compare with the true classes. Then report the successful classification rate.
   f. Compare the successful classification rate with the one given by LDA. Is it better than kernel discriminant analysis?
   g. Repeat f with QDA.

5. **Exercise 3.31.** Implement the Euler method for f being `f_oval`, `f_croissant`, and `f_sin`, in order to reproduce Figure 3.18. You will have to:

- Implement the gradient and Hessian of f. To avoid computing the analytical form, you can rely on `numDeriv::grad` and `numDeriv::hessian` to evaluate numerically $\mathrm{D}f$ and $\mathrm{H}f$ (although this will run slower than the analytical form).
- Skip initial values $\mathbf{x}_0$ such that $f(\mathbf{x}_0) < \delta$, so that the Euler method is not run for them. This saves computational time.
- Stop the iteration in the Euler method if absolute or relative convergence is attained. Precisely, stop if $\|\mathbf{x}_{t+1} - \mathbf{x}_t\|_\infty < \varepsilon$ or $\|\mathbf{x}_{t+1} - \mathbf{x}_t\|_\infty / \|\mathbf{x}_t\|_\infty < \varepsilon$. This saves computational time.
- Disregard final values $\mathbf{x}$ from the density ridge if $f(\mathbf{x}) < 50\delta$. This avoids spurious solutions.
- Join points automatically using the Euclidean minimum spanning tree algorithm.
- Tune the selection of h, N, ε, and δ. You can start with $h = 0.25$, $N = 100$, $\varepsilon = 10^{-4}$, and $\delta = 10^{-3}$.

6. **Exercise S-III.** Consider the context given in Exercise S-I. You want to automatically cluster the two apparent modes in `temps-7.txt`. To do so, you wish to rely on kernel mean shift clustering. However, it is not implemented in `ks::kms` for univariate data, so you have to code our own approach.

   a. Implement $\hat{\eta}(x;h) = \frac{h^2 \mathrm{D}\hat{f}(x;h)}{\hat{f}(x;h)}$. as an R function, named `eta_hat`, that takes as arguments a scalar `x`, a bandwidth `h`, and the vector `data`. It must return the estimated scaled gradient $\hat{\eta}(x;h)$ at the point `x`.
   b. The `eta_hat` function is slow to compute. In order to speed it, you want to define a suitable interpolating function `eta_hat_spline` obtained by calling `splinefun`. Figure out the details and check graphically that `eta_hat` and `eta_hat_spline` are equivalent.
   c. Using `eta_hat_spline`, you construct a new function, called `euler`, that implements the iterative part on the kernel mean shift algorithm. The function must take as main argument `x`, the initial point for the iterative algorithm. Consider a loop with `N = 500` iterations that is stopped if the new point in the iteration is closer than `epsilon = 1e-4` to the previous point. The function must return the final point (a point close to a mode) for which `x` has converged.
   d. Apply `euler` to all the data points in `temps-7` with a suitably-chosen bandwidth. Visualize the final points using `hist` and `rug`.
   e. On the final points resulting from d, use `kmeans` with $k = 2$ to cluster the two main modes that were detected. From there, conclude an estimation of the true proportion of defective smartphones and comment on the claim of S.

7. **Exercise 5.8.** Investigate the accuracy of the naive bootstrap confidence intervals implemented in `np::npplot`. To do so:

   1. Simulate $M = 500$ samples of size $n = 100$ from the regression model $Y = m(X) + \varepsilon$, where $m(x) = 0.25x^2 - 0.75x + 3$, $X \sim \mathcal{N}(0, 1.5^2)$, and $\varepsilon \sim \mathcal{N}(0, 0.75^2)$.
   2. Compute the 95%-confidence intervals for $m(x)$ along `x <- seq(-5, 5, by = 0.1)`, for each of the M samples. Do it for the normal approximation and quantile-based confidence intervals.
   3. Check if $m(x)$ belongs to each of the confidence intervals, for each x.
   4. Approximate the actual coverage of the confidence intervals.

   Once you have a working solution, increase n to $n = 200, 500$ and summarize your conclusions. Use $B = 500$.

8. **Exercise 5.9.** Adapt the `np_pred_CI` function to include the argument `type_boot`, which can take either the value `"naive"` or `"wild"`. If `type_boot = "wild"`, then the function must perform the wild bootstrap algorithm described above, implemented from scratch following substeps i–iv. Compare and validate the correct behavior of the confidence intervals, for the two specifications of `type_boot`, in the model considered in Exercise 5.8 (without performing the full simulation study).

9. **Exercise 5.12.** Implement your own version of the local likelihood estimator (first degree) for the Poisson regression model. To do so:

   a. Derive the local log-likelihood about x for the Poisson regression (which is analogous to (5.13)).
   b. Code from scratch an R function, `loc_pois`, that maximizes the previous local likelihood for

a vector of evaluation points. `loc_pois` must take as input the samples `X` and `Y`, the vector of evaluation points `x`, the bandwidth `h`, and the kernel `K`.

c. Implement a `cv_loc_pois` function that obtains the cross-validated bandwidth for the local Poisson regression.

d. Validate the correct behavior of `loc_pois` and `cv_loc_pois` by sampling from $Y|X = x \sim \text{Poisson}(\lambda(x))$, where $\lambda(x) = e^{\sin(x)}$ and X is distributed according to a `nor1mix::MW.nm7`.

e. Compare your results with the `locfit::locfit` function using `family = "poisson"`.