School of Mathematics and Statistics
MAST90083: Computational Statistics and Data Science
Assignment 2
Weight: 15%

Some details about Question 1 and 2

For both questions, use library "HRW" that contains the "WarsawApts" dataset. The symbol n represents length of the variables for the given dataset (WarsawApts), and a bold **1** represents vector of ones. The mean squared error (MSE) is a normalized version of residual sum of squares (RSS) and is given as MSE=RSS/n.

Notations

Throughout this assignment, small letter represents a value, small letter attached with a subscript represents an entry of a vector, small bold letter represents a vector, and capital bold letter represents a matrix.

Question 1: Splines

The aim of this question is to learn how to code linear spline basis in order to approximate a non-linear dataset via penalized spline regression (PSR). For PSR, a Cholesky decomposition (CD) and singular value decomposition (SVD) approach can be used, but for this task the regression based approach will be adopted, and any solution based on CD and SVD will not be accepted. All the questions here are connected and must be attempted sequentially.

1. Store construction.date variable from WarsawApts in $\mathbf{x}$ and areaPerMzloty in $\mathbf{y}$. Generate 20 different location of knots and store them in vector $\mathbf{k}$ (k_i corresponds to i-th value of $\mathbf{k}$, $i = 1, ..., 20$) using quantile (ref: Tutorial 5 Question 2) function. To accomplish this task, you need a numeric vector of probabilities of length 22 that can be generated using seq(0,1,length=22), however extreme values of 0 and 1 should be excluded from this function in order to match the knots and the samples in $\mathbf{x}$ (x-intercepts).

2. Using these knot locations now generate a $\mathbf{Z}$ matrix containing 20 linear spline basis functions constructed using $(\mathbf{x} - k_i)_+$. Also plot this matrix by limiting the range of y-axis between -1 and 2 as shown in Figure 1a.

3. Using the vector of ones, $\mathbf{x}$, and $\mathbf{Z}$, construct a $\mathbf{C}$ matrix of size $n \times 22$ (ref: lecture slide 18 of the spline regression). Also, generate a $\mathbf{D}$ matrix of size 22×22 consisting of ones on the diagonal entries and zeros elsewhere except the first two entries (ref: lecture slide 30 of the spline regression). Then, 100 values of tuning parameter (λ)

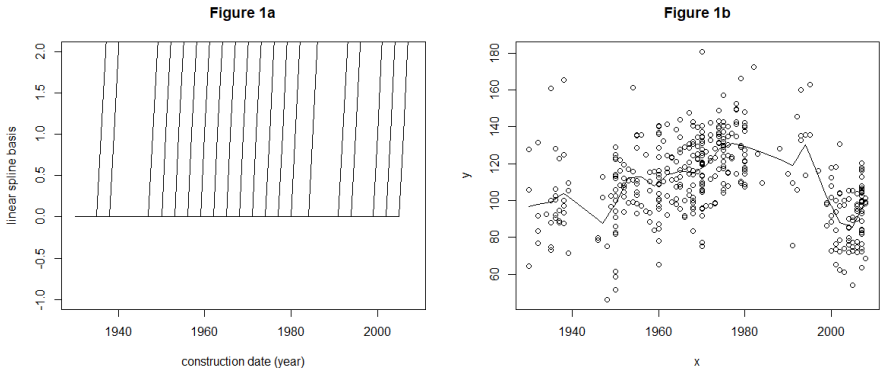

Figure 1: Spline basis and fitted curve

selected from 0 to 50 as seq(0,50,length=100) are used to estimate the fitted curves using penalized spline regression.

4. Now, compute the RSS errors, degrees of freedom, and generalized cross validation (GCV) associated with 100 tuning parameters.

5. For λ corresponding to minimum GCV, overlay the true plot with a fit of $\mathbf{y}$ as shown in Figure 1b. As you can see from the plot that with 20 linear spline basis the fitted curve looks a bit too linear and might not be a good fit. Lets increase the number of basis to 60 and obtain Figure 1c (similar to 1a) and 1d (similar to 1b). Were the results improved by increasing the number of basis? Compute the MSE to justify the improvement/deterioration.

Question 2: Non-parametric Regression

The aim of this question is to learn how to code kernel regression using different kernels. Here, question 2.1 and 2.2 are connected and must be attempted sequentially, whereas question 2.3, 2.4, and 2.5 are connected and must be attempted sequentially

1. Using 6 values of k selected from 3 to 23, write an R code for k-nearest-neighbor (NN) averaging. You can do this by firstly estimating the absolute difference between the current sample x_i and the vector $\mathbf{x}$ as $\mathbf{d} = |x_i\mathbf{1} - \mathbf{x}|$, and secondly using $\mathbf{d}$ to find those indices that correspond to its k smallest values. These indices are the indices of the k NNs to the current x_i. Use these indices to select the corresponding entries from the data vector $\mathbf{y}$ and take their mean as your i-th estimate $\hat{y}_i$.

2. For each case of k, generate a figure illustrating a fitted vector $\hat{\mathbf{y}}$ on the true vector $\mathbf{y}$ (6 sub-figures in total, must be plotted in a single figure using 3×2 subplots), and compute the MSE. For which case did you find the minimum MSE? In terms of MSE, what can be said regarding the fitting comparison between k-NN and linear spline basis?

3. Using slide 13 of the kernel regression lecture, write a single function to accommodate all 6 kernels. This function must be capable of returning 6 values, where each value corresponds to the output value of a single kernel.

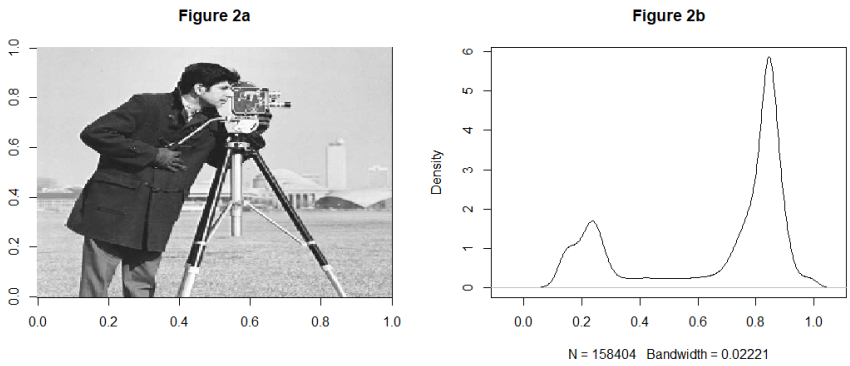

Figure 2: Cameraman picture and its density plot

4. Keeping the bandwidth fixed at $h = 2$ and using formula $\hat{f}(x_i) = \frac{\sum_{j=1}^{n} \mathbf{K}_h(x_i,x_j) y_j}{\sum_{j=1}^{n} \mathbf{K}_h(x_i,x_j)}$, (where $\{j = 1, ..., n\}$) code 6 kernel regression based fits for $\mathbf{y}$ by using 6 kernels from the previous question (To elaborate, for each sample x_i and x_j, six different kernel values must be obtained in a single go to eventually generate six estimated values of the i-th fit $\hat{\mathbf{f}}(x_i)$, where $\hat{\mathbf{f}}(x_i)$ is a vector and has 6 entries).

5. For each of the six kernels, generate a figure illustrating fitted vector $\hat{f}(x)$ on the true vector $\mathbf{y}$ (6 sub-figures in total, must be plotted in a single figure using 3×2 subplots). Also, for each kernel estimate the MSE. For which kernel, the MSE is found to be minimum?

Question 3: Expectation-Maximization

For this question, you are not allowed to use R's built in function for the expectation-maximization (EM) algorithm. You have already learnt how to code EM algorithm for a gaussian mixture consisting of two probability density functions (pdfs), where both had same standard deviation. Here, you will extend that algorithm to three cases, where all three having different standard deviations.

1. In order to load relevant libraries and the provided image that is needed for the subsequent questions, use the following code

```
library("plot.matrix")
library("png")
library("fields")
I <- readPNG("D:/Codefiles/CM.png")
I=I[,,1]
I=t(apply(I, 2, rev))
par(mfrow=c(2,1))
image(I, col = gray((0:255)/255))
plot(density(I))
```

2. The above code will generate figure 2 and as you can see from the density plot of the cameraman picture (Figure 2b) that it is a mixture of three Gaussian distributions with different standard deviations for each. Therefore, extend the EM algorithm so that it can estimate parameters for these three pdfs. You may consider reshaping the image data matrix $\mathbf{I}$ to one dimensional vector before applying your algorithm.

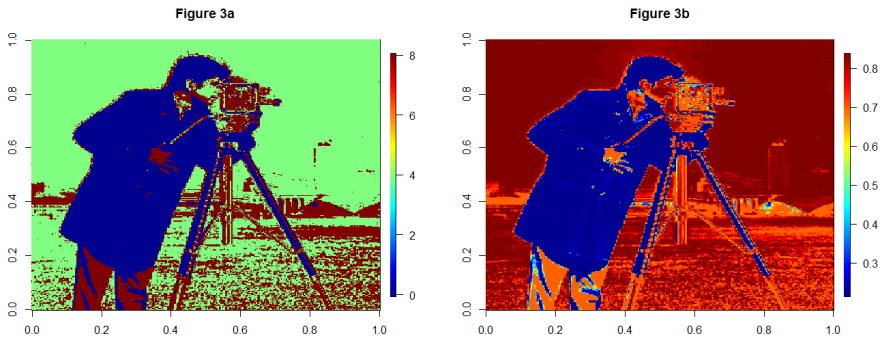

Figure 3: Cameraman picture labeling

3. You must terminate the algorithm when the stopping criteria $(e_j - e_{j-1})$ becomes less than 10^{-6}, where $e_j = \sum_{i=1}^{3}(m_i - m_i^*) + \sum(c_i - c_i^*)$. Here, e_j and e_{j-1} correspond to the values from the current and previous iteration, respectively. Furthermore, m_i and c_i represent i-th mean and standard deviation from the current iteration, respectively, whereas m_i^* and c_i^* represent i-th mean and standard deviation from the previous iteration. Also, use command "print(round(c($\mathbf{m}$, $\mathbf{c}$, $\mathbf{p}$), 4))" to print all 9 values in each iteration, where p_i (the i-th entry of vector $\mathbf{p}$) is a mixing probability for the i-th distribution and denoted as $\hat{\pi}$ in the lecture slide 24 of the missing data and EM algorithm.

4. Using the results of EM algorithm, label the image pixels by using i) each pixel's pdf estimation (you must be able to read the assignment value from figure 3a, you may want to use "dnorm" function for this part), and ii) posterior pdfs multiplied with their mean.

5. Plot both images obtained from the last question, and compare the results with Figure 3. You must obtain an exact match. You may want to use function "image.plot" for better visualization.