Biostatistics 615 Mastery Assignment #4 (10 pts)

Due by October 8th 2024 (Tuesday) 11:59pm. Use Gradescope (via Canvas) to submit an R file.

- Your submission should only contain one `R` file named `kernelRidgeRegression.R` that contains a function named `kernelRidgeRegression(X, Y, lambda)`.
- Your code will be evaluated in Gradescope using 10 different test cases using an automated script. Full credit will be given if your code passes all test cases.
- You are allowed to submit multiple times before the deadline, but only the last submission will be graded. Automated feedback will be provided for each submission.
- You need to implement the function to work with arbitrary (valid) input values beyond the 10 cases tested. If you tweak your implementation so that your functions works specifically for the test cases, you will not receive any credit.
- You may test the function in the Google Colab page at https://bit.ly/615hw4xtra to test your code on a subset of test cases.
- Implement your function as efficient as you can. If your program does not finish after running for 3.0 seconds, you will lose the points for those test cases. Note that the official solution finishes much faster for any test case, so this should be a reasonable time limit. Also, note that the running time includes the time for reading the input files. The time reported in the Google Colab test page will be much faster because the input files are already loaded in the memory.
- All the returned values should have at least 8 correct significant digits, so make sure that your implementation retains sufficient numerical precision.

Problem 1 - `kernelRidgeRegression.R` (10 pts)

Suppose $\{X_i, Y_i\}_{i=1}^n$ are the training dataset where X_i and Y_i are both scalar. To model the relationship between Y_i and X_i, we consider a function f in the reproducing kernel Hilbert space, i.e.

$$f(x; \boldsymbol{\beta}) = \sum_{j=1}^{n} \beta_j \kappa(x, X_j),$$

where $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_n)^T$ and κ is a truncated kernel defined as

$$\kappa(x, x') = \begin{cases} \exp\{-\rho(x - x')^2\}, & |x - x'| \leq h \\ 0, & |x - x'| > h \end{cases}$$

where the parameter $\rho (\rho > 0)$ controls the smoothness of the kernel and the parameter $h (h > 0)$ is the bandwidth.

To estimate $\boldsymbol{\beta}$, we solve the following optimization problem.

$$\min_{\boldsymbol{\beta}} \left\{ \sum_{i=1}^{n} \{Y_i - f(X_i; \boldsymbol{\beta})\}^2 + \lambda \boldsymbol{\beta}^T \boldsymbol{K} \boldsymbol{\beta} \right\},$$

where $\lambda(\lambda > 0)$ is a pre-specified tuning parameter and $\boldsymbol{K} = \{\kappa(X_i, X_j)\}_{n\times n}$. The solution $\hat{\boldsymbol{\beta}} = (\hat{\beta}_1, \ldots, \hat{\beta}_n)^T$ satisfies the following linear equations:

$$(\boldsymbol{K} + \lambda \boldsymbol{I})\hat{\boldsymbol{\beta}} = \boldsymbol{Y},$$

where $\boldsymbol{Y} = (Y_1, \ldots, Y_n)^T$. We evaluate the performance of the model fitting on the test dataset $\{X_i^*, Y_i^*\}_{i=1}^m$ by computing the predictive mean square errors:

$$\text{PMSE} = \frac{1}{m}\sum_{i=1}^{m}\left\{Y_i^* - f(X_i^*; \hat{\boldsymbol{\beta}})\right\}^2.$$

Your task is to implement a function `kernelRidgeRegression(df, lambda, rho, bw)` that takes the following input:

1. `df`: a data frame that contains 4 columns – `x_train`, `y_train`, `x_test`, `x_test`
2. `lambda`: the tuning parameter λ
3. `rho`: the smoothness parameter ρ
4. `bw`: the bandwidth parameter h

The function should return a numeric value of the predictive mean square errors (PMSE).

In the following running example, the function prints first 5 predicted values so that you can check whether the function is implemented correctly (you do not have to print the predicted values in your implementation).

```
> dat = read.table("data/test.1.tsv", header=TRUE)
> params = scan("data/test.1.args", quiet=TRUE)
> pmse <- kernelRidgeRegression(dat, params[1], params[2], params[3])
beta[1:5]:
0.6974414
-0.8825270
-0.0483076
0.0669788
0.6364501

y_pred[1:5]:
3.8046238
8.4307273
-9.7743867
4.9784528
-10.103766
> print(pmse)
[1] 8.029016
```

The example input files "`test.1.tsv`" and "`test.1.args`" are provided within the file `hw4_xtra_examples.tar.gz` accessible in https://bit.ly/615hw4xtra.

The actual examples that will be used for grading will contain more complex examples, so be sure to implement your function to work with arbitrary valid input values.

As noted above, for this problem, you are allowed to use the `Matrix` library. Note that the `Matrix` library is already loaded, and you should NOT include `library(Matrix)` in your code. Calling `library(...)` in your submitted cause will cause errors. Note that you are NOT allowed to use any functions outside the `base` and `Matrix` package in your implementation. Use `help(...)` to check whether a function belongs to the `base` or `Matrix` package or not. Using `sparseMatrix()` function in the `Matrix` package is recommended when constructing a sparse matrix.

The cloud machine that will be used for grading has a very small (0.5GB) memory, so make sure that you do not use too much memory in your implementation. If you code runs out of memory, "Killed" error messages will appear in the user output.

No error handling for malformed arguments is needed.