STAT 542 / CS 598: Homework 1

Fall 2019, by Ruoqing Zhu (rqzhu)

Due: Monday, Sep 9 by 11:59 PM Pacific Time

Contents

Directions

Students are encouraged to work together on homework. However, sharing, copying, or providing any part of a homework solution or code is an infraction of the University's rules on Academic Integrity. Any violation will be punished as severely as possible. Final submissions must be uploaded to your homework submission portal. No email or hardcopy will be accepted. For late submission policy and grading rubrics, please refer to the course website.

- You are required to submit two files:
  - Your `.rmd` RMarkdown (or Python) file which should be saved as `HW1_yourNetID.Rmd`. For example, `HW1_rqzhu.Rmd`.
  - The result of knitting your RMarkdown file as `HW1_yourNetID.pdf`. For example, `HW1_rqzhu.pdf`. Please note that this must be a `.pdf` file. `.html` format cannot be accepted.
- Your resulting `.pdf` file will be considered as a report, which is the material that will determine the majority of your grade. Be sure to visibly include all `R` code and output that is relevant to answering the exercises.
- If you use the example homework `.Rmd` file (provided here) as a template, be sure to remove the directions section.
- Your `.Rmd` file should be written such that, if it is placed in a folder with any data you are asked to import, it will knit properly without modification.
- Include your Name and NetID in your report.
- **Late policy**: You can also choose to submit your assignment as late as four days after the deadline, which is 11:30 PM on Monday in the following week. However, **you will lose 10% of your score**.

Question 1 [50 Points] KNN

Write an R function to fit a KNN regression model. Complete the following steps

a. [15 Points] Write a function `myknn(xtest, xtrain, ytrain, k)` that fits a KNN model that predict a target point or multiple target points `xtest`. Here `xtrain` is the training dataset covariate value, `ytrain` is the training data outcome, and `k` is the number of nearest neighbors. Use the ℓ_2 norm to evaluate the distance between two points. Please note that you cannot use any additional `R` package within this function.

b. [10 Points] Generate 1000 observations from a five-dimensional normally distribution:

$$\mathcal{N}(\mu, \Sigma_{5\times 5})$$

where $\mu = (1, 2, 3, 4, 5)^{\mathrm{T}}$ and $\Sigma_{5\times 5}$ is an autoregressive covariance matrix, with the (i, j)th entry equal to $0.5^{|i-j|}$. Then, generate outcome values Y based on the linear model

$$Y = X_1 + X_2 + (X_3 - 2.5)^2 + \epsilon$$

where ϵ follows i.i.d. standard normal distribution. Use `set.seed(1)` right before you generate this entire data. Print the first 3 entries of your data.

c. [10 Points] Use the first 400 observations of your data as the training data and the rest as testing data. Predict the Y values using your KNN function with `k = 5`. Evaluate the prediction accuracy using mean squared error

$$\frac{1}{N}\sum_i (y_i - \widehat{y}_i)^2$$

d. [15 Points] Compare the prediction error of a linear model with your KNN model. Consider k being 1, 2, 3, ..., 9, 10, 15, 20, ..., 95, 100. Demonstrate all results in a single, easily interpretable figure with proper legends.

Question 2 [50 Points] Linear Regression through Optimization

Linear regression is most popular statistical model, and the core technique for solving a linear regression is simply inverting a matrix:

$$\widehat{\boldsymbol{\beta}} = \left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}$$

However, lets consider alternative approaches to solve linear regression through optimization. We use a gradient descent approach. We know that $\widehat{\boldsymbol{\beta}}$ can also be expressed as

$$\widehat{\boldsymbol{\beta}} = \arg\min \ell(\boldsymbol{\beta}) = \arg\min \frac{1}{2n}\sum_{i=1}^{n}(y_i - x_i^{\mathrm{T}}\boldsymbol{\beta})^2.$$

And the gradient can be derived

$$\frac{\partial \ell(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = -\frac{1}{n}\sum_{i=1}^{n}(y_i - x_i^{\mathrm{T}}\boldsymbol{\beta})x_i.$$

To perform the optimization, we will first set an initial beta value, say $\boldsymbol{\beta} = \mathbf{0}$ for all entries, then proceed with the updating

$$\boldsymbol{\beta}^{\mathrm{new}} = \boldsymbol{\beta}^{\mathrm{old}} - \frac{\partial \ell(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} \times \delta,$$

where δ is some small constant, say 0.1. We will keep updating the beta values by setting $\boldsymbol{\beta}^{\mathrm{new}}$ as the old value and calcuting a new one untill the difference between $\boldsymbol{\beta}^{\mathrm{new}}$ and $\boldsymbol{\beta}^{\mathrm{old}}$ is less than a prespecified threshold ϵ, e.g., $\epsilon = 10^{-6}$. You should also set a maximum number of iterations to prevent excessively long runing time.

a. [35 Points] Based on this description, write your own `R` function `mylm_g(x, y, delta, epsilon, maxitr)` to implement this optimization version of linear regression. The output of this function should be a vector of the estimated beta value.

b. [15 Points] Test this function on the Boston Housing data from the `mlbench` package. Documentation is provided here if you need a description of the data. We will remove `medv`, `town` and `tract` from the data and use `cmedv` as the outcome. We will use a scaled and centered version of the data for estimation. Please also note that in this case, you do not need the intercept term. And you should compare your result to the `lm()` function on the same data. Experiment on different `maxitr` values to obtain a good solution. However your function should not run more than a few seconds.

```
library(mlbench)
data(BostonHousing2)
X = BostonHousing2[, !(colnames(BostonHousing2) %in% c("medv", "town", "tract", "cmedv"))]
X = data.matrix(X)
X = scale(X)
Y = as.vector(scale(BostonHousing2$cmedv))
```

Bonus Question [5 Points] The Non-scaled Version

When we do not scale and center the data matrix (both X and Y), it could be challenging to obtain a good solution. Try this with your code, and comment on what you observed and explain why. Can you think of a way to calculate the beta parameters on the original scale using the solution from the previous question? To earn a full 5 point bonus, you must provide a rigors mathematical derivation and also validate that by comparing it to the `lm()` function on the original data.