

BIOS 731: Advanced Statistical Computing

Fall 2020 Homework 3

Due 10/8/2022 Saturday at 11:59pm

Instruction:

- Please submit both write-ups and programs in two separate files.
- All submissions need to be in electronic format.
- The write-up is preferred be in pdf format (written in Word or L^AT_EX, or scanned hand-written document). You can take a picture of hand-written document and submit JPG if you don't have a scanner, but make sure the picture is clear and readable. Name the file **BIOS731_NAME_hw3.EXT**. Replace NAME by your name, and EXT by proper extension name (pdf or jpg).
- The programs need to be written in a high-level language (no compilation required), and R is highly recommended. The codes for all problems need to be saved in a **single** file named **BIOS731_NAME_hw3.EXT**. Replace NAME by your name, and EXT by proper extension name, e.g., R, sas, m, py, etc. Provide adequate comments in the codes to clearly mark the section for different questions. The codes should generate all results and figures in the homework. Please make sure the codes are “self-contained”, e.g., does not depend on platform, can be run at any other machine in any subdirectory, and does not require user input.
- Total is 100 points.

Problem 1: HMM (60 points)

Hidden Markov model (HMM) is useful for modeling financial time series data such as the stock prices. In this homework we will practice using HMM to model the daily price of a hypothetical stock.

Define the closing price for the stock at day t is $x_t, t = 0, \dots, T$, and the “log returns” as $r_t = \log(x_t/x_{t-1}), t = 1, \dots, T$. We can assume the market for day t belongs to one of the 3 states: “bullish”, “bearish” and “flat”, which means that the prices are going up, going down or fluctuating. Denote the state of day t by Z_t , the daily log return can be modeled by following 3-state HMM with Normal emission probability:

$$Pr(Z_1 = k) = \pi_k$$

$$Pr(Z_{t+1} = l | Z_t = k) = P_{kl}$$

$$r_t | Z_t = k \sim N(\mu_k, \sigma_k^2), k = 1, 2, 3$$

Obtain the simulated price data from the class website, then answer following questions:

1. Why is it important to use log returns instead of the daily price in the HMM? Can we formulate a HMM using the prices? **Hint:** check whether the data satisfy the HMM model assumptions. (10 points)
2. The parameters for a 3-state HMM are $(\pi_k, P_{kl}, \mu_k, \sigma_k)$. Forward-backward algorithm with EM can be applied to estimate these parameters iteratively. At iteration i ,
 - (a) Given current values of the parameters, write down the expressions of forward and backward probabilities. (10 points)
 - (b) Give the forward and backward probabilities, write down the procedures for estimating model parameters. (10 points)
3. Implement the forward and backward algorithm with EM in a programming language of your choice, and report the estimates of model parameters. (20 points)
4. Assume we want to predict tomorrow's stock price based on our HMM results. What's the marginal distribution of x_{T+1} given x_1, \dots, x_T ? What are its mean and variance? **Hint:** if you cannot derive the closed form solution, use a simulation to obtain these values. (10 points)

Disclaimer: the stock price is simulated and this homework is for training purpose only. Please do NOT attempt to apply this to real world trading. The modeling of actual financial data is far more complicated.

Problem 2: Median Regression (20 points)

Use linear programming to estimate the coefficients for a quantile regression. You need to write a function named “myrq”, which takes a response vector \mathbf{y} , a covariate matrix \mathbf{X} and quantile τ ($\tau = .5$ for median regression). Existing linear programming functions can be used directly to solve the LP problem (For example, `simplex` function in `boot` package, or `lp` function in `lpSolve` package). The function should return the estimated coefficient for the corresponding quantile regression. Following codes could provide test data and median regression results to verify your own function.

```
library(quantreg)
data(stackloss)
rq(stack.loss ~ stack.x, .5)
```

Problem 3: Implementation of LASSO (20 points)

As illustrated in class, a LASSO problem can be rewritten as a quadratic programming problem:

$$\begin{aligned} \max \quad & - \sum_{i=1}^n (y_i - \sum_j b_j^+ x_j + \sum_j b_j^- x_j)^2 \\ \text{s.t.} \quad & \sum_j (b_j^+ + b_j^-) \leq t, \\ & b_j^+, b_j^- \geq 0 \end{aligned}$$

This is a standard QP problem which can be solved by QP solvers.

1. Many widely used QP solvers require the matrix in the quadratic function for second order term to be positive definite (such as `solve.QP` in `quadprog` package). Rewrite the quadratic programming problem for LASSO in matrix form and show that the matrix is not positive definite, thus QP solvers like `solve.QP` cannot be used. (10 points)
2. `LowRankQP` function in `LowRankQP` package can handle non positive definite situation. Use the matrix format you derived in previous part and `LowRankQP` to write your own function ‘`myLasso`’ to estimate the coefficient for a LASSO problem. Your function needs to take three parameters: `Y` (response), `X` (predictor), and `t` (tuning parameter). (10 points)

Following codes can be used to verify your own function. Note that the results are not exactly the same, because the objective functions are constructed differently (Actually ‘`glmnet`’ and ‘`lars`’ provide different estimates). However the trend (variables got selected/deselected) will be similar.

```

### Obtain diabetes data
library(lars)
data(diabetes)

### Lasso results from R package "glmnet"
library(glmnet)
res = glmnet(diabetes$x,diabetes$y)
plot(res, "lambda")
cv = cv.glmnet(diabetes$x,diabetes$y)
plot(cv)
coef(res, s=5)

### Lasso results from R package "lars"
object <- lars(diabetes$x,diabetes$y)
plot(object)
object$lambda
object$beta
object$beta[6,] ## this one has similar results as glmnet with s=5

```