BIOS 731: Advanced Statistical Computing

Fall 2016 Homework 2

Due 10/13/2016 at 4pm before the class

Instruction: Please submit both write-ups and programs. 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 NAME_hw1.EXT. Replace NAME by your name, and EXT by proper extension name, e.g., R, sas, 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.

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 hypothetic stock.

Define the closing price for the stock at day t is $x_t, t = 0, \ldots, T$, and the "log returns" as $r_t = log(x_t/x_{t-1}), t = 1, \ldots, T$. 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:

$$
\begin{aligned}
&Pr(Z_1 = k) = \pi_k \\
&Pr(Z_{t+1} = l | Z_t = k) = P_{kl} \\
&r_t | Z_t = k \sim N(\mu_k, \sigma_k^2), k = 1, 2, 3
\end{aligned}
$$

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**: consider the assumptions of a homogeneous Markov chain, then check whether the data satisfy these 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 $x_1, \ldots, x_T$? What are its mean and variance? **Hint**: if you cannot derive the closed form solution, use a simulation to obtain these values. (10 points)

Computational tip: When the HMM chain is long, the computation of forward/backward matrices has to be done in logarithm scale, otherwise they will become 0 very quickly. However to evaluate the sum in log-scale is tricky, for example, $\log(e^a + e^b)$ will become negative infinity when a or b are negative number with large absolute values. Use the following trick to deal with the scenario:

$$log(e^a + e^b) = log(e^a(1 + e^{b-a})) = a + log(1 + e^{b-a})$$

It equals b when $b >> a$, equals a when $b << a$. When the values of b and a are close, the computation is numerically stable. Following is an R implementation of the algorithm, which works for two vectors:

```
Raddlog <- function (a, b)
{
    result <- rep(0, length(a))
    idx1 <- a > b + 200
    result[idx1] <- a[idx1]
    idx2 <- b > a + 200
    result[idx2] <- b[idx2]
    idx0 <- !(idx1 | idx2)
    result[idx0] <- a[idx0] + log1p(exp(b[idx0] - a[idx0]))
    result
}
```

A simple test (with a and b being -1000) shows that they work well, whereas directly summing up then taking log doesn't work:

```
> Raddlog(-1000, -1000)
[1] -999.3069
> log(exp(-1000)+exp(-1000))
[1] -Inf
```

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 `y`, a covariate matrix `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:

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

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 could provide test data and Lasso results 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
```