MA 589 — Computational Statistics

Project 2

(Due: Friday, 10/12/18)

1. Suppose you have a polynomial $p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$ and you wish to evaluate it at x^*. Instead of simply plugging-in x^* in the expression for p, you could obtain $p(x^*)$ more efficiently using *Horner's scheme*. The following `R` function takes a vector with coefficients `c(a0, a1, ..., an)` and returns a function[1] that takes x^* as argument and returns $p(x^*)$.

```
horner <- function (coef)
  function (x) {
    s <- 0
    for (i in seq(length(coef), 1, -1))
      s <- s * x + coef[i]
    s
  }
```

   (a) Explain why `horner` works; start by understanding how a small polynomial, say `c(3, -2, 1)`, is evaluated, and then provide a mathematical expression that summarizes how the computations are performed.

   (b) Use `horner` to plot $p(x) = 2 - 4x - x^2/2 + x^3$ for $x \in [-3, 3]$. (Hint: check the code in the next problem, or see `curve` in `R`.)

   (c) Implement Newton's method to find the roots of p from the previous item. Note that since the derivative of p is also a polynomial, you can still use `horner` to evaluate it[2]. What roots do you find when starting at $x^{(0)} = -1.5$, $x^{(0)} = 0$, and $x^{(0)} = 1.5$? What happens when you start at $x^{(0)} = -1$?

   (d) The *Legendre* polynomial Le_n of order n is defined as

$$\mathrm{Le}_n(x) = \frac{1}{2^n n!} \frac{\mathrm{d}^n}{\mathrm{d}x^n} \left(x^2 - 1\right)^n.$$

   For instance, $\mathrm{Le}_0(x) = 1$ and $\mathrm{Le}_1(x) = x$, and we can derive the recurrent formulation $(n+1)\mathrm{Le}_{n+1}(x) = (2n+1)x\mathrm{Le}_n(x) - n\mathrm{Le}_{n-1}(x)$. Using these facts, write a function that returns the coefficients of Le_n to be used by `horner`. $(*)$ Your function should exploit *memoization*, that is, it should cache the coefficients from previous calls to save computing time.

2. A friend of yours wants to understand the behavior of a variable y as a function of another variable x and so collects the following data:

[1] That is, `horner` is a *higher order function*.

[2] Can you come up with a `R` function that returns the coefficients of p'?

i	1	2	3	4	5	6	7	8	9	10	11	12	13
x	-3.0	-2.5	-2.0	-1.5	-1.0	-0.5	0.0	0.5	1.0	1.5	2.0	2.5	3.0
y	-17.0	-7.0	0.5	3.0	5.0	4.0	2.5	-0.5	-2.0	-2.0	0.5	5.0	12.0

Your friend knows that $y = p(x)$ where p is a polynomial of (hopefully) low degree. You suggest him to fit a regression assuming that the degree is four to find the coefficients β:

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \beta_4 x_i^4, \quad i = 1, \ldots, 13.$$

(a) Build matrices for X—this is not simply x, but the "predictors" in the expression above; it is a *Vandermonde* matrix—and y. Now find $X^T X$ and $X^T y$ and assemble the tableau

$$T = \begin{bmatrix} X^T X & X^T y \\ y^T X & y^T y \end{bmatrix}.$$

(b) Implement the sweep operator: create a function `SWEEP` [3] that takes a tableau T and a row k in T and returns an updated tableau with its k-th row swept. Quick check: what is $\texttt{SWEEP}(\texttt{SWEEP}(T, k), k)$ for $k = 1, \ldots, 5$?

(c) Every time we sweep a row that corresponds to a predictor, we include it in the regression (or exclude it if it is already in.) Having this in mind, how do you explain the output from the following code? What is being plotted at each `k`? What is being printed at each `k`?

```
plot(x, y)
a <- seq(-3, 3, length.out = 100)
p <- ncol(T) - 1
for (k in 1:p) {
  T <- SWEEP(T, k)
  lines(a, horner(T[1:k, p + 1])(a), lty = k)
  print(c(k, T[p + 1, p + 1]))
}
```

(d) (∗) Your friend believes that p has roots at -2 and 2, and so wishes to constrain β by requiring $L\beta = 0$ where

$$L = \begin{bmatrix} 1 & -2 & 4 & -8 & 16 \\ 1 & 2 & 4 & 8 & 16 \end{bmatrix}.$$

To do that, you further extend T to accommodate the new requirement,

$$T = \begin{bmatrix} X^T X & L^T & X^T y \\ L & 0 & 0 \\ y^T X & 0 & y^T y \end{bmatrix},$$

[3] Unfortunately, `R` already has a `sweep` function, and it does not correspond to this operator.

since sweeping the two rows that correspond to L will solve $L\beta = 0$. What is the constrained solution $\hat{\beta}$ you obtain after sweeping these two rows[4]? Check that $L\hat{\beta} = 0$ (within machine precision, of course) and plot the polynomial with $\hat{\beta}$ as coefficients (just add it to the previous plot.)

Comment on this result; for instance, is it much different from the unconstrained solution? How would you quantify this difference in terms of fit to the data[5]?

3. The SWEEP operator can also be useful when deriving matrix identities. Suppose we have normal random vectors $\mathbf{x}$ and $\mathbf{y}$ with joint distribution

$$\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix} \sim N\left(\begin{bmatrix}\mu_x\\ \mu_y\end{bmatrix}, \begin{bmatrix}\Sigma_{xx} & \Sigma_{xy}\\ \Sigma_{xy}^\top & \Sigma_{yy}\end{bmatrix}\right).$$

To obtain the inverse of the joint variance matrix of $\mathbf{x}$ and $\mathbf{y}$ we can apply the SWEEP operator in two steps:

$$\Sigma = \begin{bmatrix}\Sigma_{xx} & \Sigma_{xy}\\ \Sigma_{xy}^\top & \Sigma_{yy}\end{bmatrix} \xleftrightarrow[\text{xx-block}]{SWEEP} V := \begin{bmatrix}{\Sigma_{xx}}^{-1} & {\Sigma_{xx}}^{-1}\Sigma_{xy}\\ -\Sigma_{xy}^\top{\Sigma_{xx}}^{-1} & S := \Sigma_{yy} - \Sigma_{xy}^\top{\Sigma_{xx}}^{-1}\Sigma_{xy}\end{bmatrix}$$
$$\xleftrightarrow[\text{yy-block}]{SWEEP} \begin{bmatrix}{\Sigma_{xx}}^{-1} + {\Sigma_{xx}}^{-1}\Sigma_{xy}S^{-1}\Sigma_{xy}^\top{\Sigma_{xx}}^{-1} & -{\Sigma_{xx}}^{-1}\Sigma_{xy}S^{-1}\\ -S^{-1}\Sigma_{xy}^\top{\Sigma_{xx}}^{-1} & S^{-1}\end{bmatrix} = \Sigma^{-1}.$$

The yy-block of V, defined as matrix S, is the *Schur complement* of Σ_{yy} in Σ, denoted $\mathcal{SC}_{yy}(\Sigma)$.

(a) Perform the first SWEEP step to verify that $(\Sigma^{-1})_{yy}$, the yy-block of Σ^{-1} is the inverse Schur complement of Σ_{yy}, $\mathcal{SC}_{yy}(\Sigma)^{-1}$ (and, thus, Schur complements make block and inverse operations almost commutable).

(b) Similarly, show that, after the second SWEEP step,

$$(\Sigma^{-1})_{xx} = {\Sigma_{xx}}^{-1} + {\Sigma_{xx}}^{-1}\Sigma_{xy}S^{-1}\Sigma_{xy}^\top{\Sigma_{xx}}^{-1},$$

and that, by a symmetry argument when applying a SWEEP step to the yy-block of Σ, that $(\Sigma^{-1})_{xx} = (\Sigma_{xx} - \Sigma_{xy}{\Sigma_{yy}}^{-1}\Sigma_{xy}^\top)^{-1}$. This is a symmetric version of the *Woodbury* identity[6].

(c) It can be shown that

$$\mathbf{y} \,|\, \mathbf{x} \sim N\Big(\mu_y + \Sigma_{xy}^\top\Sigma_{xx}^{-1}(\mathbf{x} - \mu_x), \Sigma_{yy} - \Sigma_{xy}^\top\Sigma_{xx}^{-1}\Sigma_{yx}\Big).$$

Show how you would assemble a tableau to compute the conditional mean and variance of $\mathbf{y}$ given $\mathbf{x}$ using the SWEEP operator.

---

[4] Aren't you curious about what is in block $(2,3)$ of T after the sweeps?

[5] If you're feeling adventurous you can even formally conduct a F-test.

[6] In fact, we can derive the general Woodbury identity by following a similar argument on a non-symmetric matrix.