Quantitative Genomics and Genetics 2016
Computer Lab 10

Jin Hyun Ju (jj328@cornell.edu)

April 28, 2016

1 Linear Mixed Models

Linear mixed models are named so because unlike linear models where we only used fixed effects to model the dependent variables, they use a "mix" of fixed effects and random effects. To quickly review the linear model to directly compare their difference let's go back to the simplest form we have used to model genotype effects on the phenotype.

$$\vec{y_i} = \mathbf{X}\beta_x + \vec{\epsilon}$$

$$\vec{\epsilon} \sim N(0, \sigma_e^2 \mathbf{I})$$

$\vec{y_i}$ is a vector with n measurements of the phenotype, $\mathbf{X}$ is the $n \times j$ matrix with $j-1$ indenpendent variables and one column with 1s for the mean, and $\vec{\epsilon}$ is the normally distributed error with a variance of σ_e^2. We can also represent the model focusing on the normal error. The model can be shown as:

$$\vec{y_i} \sim N(\mathbf{X}\vec{\beta}, \sigma^2 \mathbf{I})$$

2 EM algorithm

I am including a mini exercise about the EM algorithm for those of you who are interested. In class we learned about mixed models and the EM algorithm to estimate the parameter values for the mixed model. The process of the EM algorithm is outlined below. Basically, what happens is that it fixes the parameters (betas, and sigmas) for the first step to calculate the best estimate for a and Va and in the next step it uses the values calculated for a and Va to update the betas and the sigmas until the values don't change much (reach convergence).

The actual R code to run an EM algorithm on the dataset that is posted on the website is shown below. Here are some mini tasks regarding this exercise:

Take a look at the given data and inspect the dimensions of each.

For the given code, include a line that prints out the log likelihood for each iteration (Ex. $iteration = 1$, $loglikelihood = 500.123$). Also plot the log likelihood for each iteration in a line. What does it look like?

Compare the results of the EM algorithm to the results of a simple linear fixed effect model and see what is different.

```
library(MASS)  # load MASS package to use the ginv() function

X = as.matrix(read.table("QG16_Lab11_EM_X.txt"))
Y = as.matrix(read.table("QG16_Lab11_EM_Y.txt"))
A = as.matrix(read.table("QG16_Lab11_EM_A.txt"))



EM_algorithm = function(Y, X_j, A) {
    # Calculate the inverse of A once since it is used repeatedly in the
    # algorithm
    solve_A = ginv(A)

    n = length(Y)

    I = diag(1, n)

    log_L = c()
    # set starting values
    sigma_sq_a = 70
    sigma_sq_e = 10
    beta = as.vector(rep(0, ncol(X_j)))


    C = A * sigma_sq_a + I * sigma_sq_e
    log_L[1] = -1/2 * determinant(C)$modulus - 1/2 * t(Y - X_j %*% beta) %*%
        ginv(C) %*% (Y - X_j %*% beta)
    iter = 2

    while (1) {

        S = ginv(I + solve_A * sigma_sq_e/sigma_sq_a)

        alpha = S %*% (Y - X_j %*% beta)

        V = S * sigma_sq_e

        beta = ginv(t(X_j) %*% X_j) %*% t(X_j) %*% (Y - alpha)
```

```
        sigma_sq_a = 1/n * (t(alpha) %*% solve_A %*% alpha + sum(diag(solve_A %*%
            V)))

        sigma_sq_e = 1/n * (t(Y - X_j %*% beta - alpha) %*% (Y - X_j %*% beta -
            alpha) + sum(diag(V)))

        C = A * sigma_sq_a + I * sigma_sq_e
        log_L[iter] = -1/2 * determinant(C)$modulus - 1/2 * t(Y - X_j %*% beta) %*%
            ginv(C) %*% (Y - X_j %*% beta)

        if (log_L[iter] - log_L[iter - 1] < 1e-05) {
            break
        }

        iter = iter + 1
    }

    return(list(beta = beta, sigma_sq_a = sigma_sq_a, sigma_sq_e = sigma_sq_e,
        log_L = log_L[iter - 1]))

}



################ Mixed model #

n_indivs = length(Y)

# Null model
One = as.matrix(rep(1, n_indivs))
log_L_null = EM_algorithm(Y, One, A)$log_L

## Error in A * sigma_sq_a:  non-conformable arrays

p_values_EM = c()

# Full model
for (j in 1:ncol(X)) {

    X_j = cbind(1, X[, j])

    fit = EM_algorithm(Y, X_j, A)

    p_values_EM[j] = pchisq(-2 * (log_L_null - fit$log_L), 1, lower.tail = FALSE)
```

```
    cat(".")
}
```

```
## Error in A * sigma_sq_a:  non-conformable arrays
```