QT_TDS_report

April 30, 2015

Paper: Quantitative trait analysis in sequencing studies under trait-dependent sampling.
Authors: Dan-Yu Lin, Donglin Zeng, and Zheng-Zheng Tang.

1 Background

Due to the high cost of whole-genome sequencing, a cost-effective strategy has been to select subjects with extreme quantitative trait values for. If such trait-dependent sampling is accounted for properly, it proves to substantially increase statistical power. The conventional approach of a standard linear regression retains its type I error rate in the absence of genetic association. However, the standard linear regression will yield biased estimates of the genetic effects in the present of genetic association. Furthermore, to gain higher statistical power, the authors combine the data of multiple studies that have common quantitative traits to have a larger sample size. Consequently, the authors propose an efficient-likelihood based methods for analyzing primary and secondary quantitative traits under trait-dependent sampling. The new approached is named maximum likelihood estimation (MLE). In this project, I will implement this method in Python and apply the algorithm to a simulated dataset that mimicks the structure of the simulation studies constructed in the paper reference paper. For the purpose of this final project, I want to focus the likelihood estimation of the primary and secondary traits through the use of EM algorithm.

From the simulation studies done in the reference paper, this MLE approach has well-controlled type I error rate and high power in detecting the genetic effects if presence. The nature of the problem which motivates the development of this method is that we have a phenotype of interest that is well-measured in a study. This particular phenotype is a covariate in other studies. We want to be able to combine the data from these studies together to increase sample size. However, the down side of a large study would be the cost of sequencing. Thus, we only sequence subjects with extreme phenotypic values. Under such framework, the new MLE method would be most efficient and unbiasedly estimates the genetic effects. This method would be most useful for meta-analysis in research areas where multiple studies measure the same phenotype of interest.

2 Methods

For a study, we denote the primary quantitative trait Y_1 and the secondary quantitative trait Y_2. G_1 and G_2 represent the genetic variables for each of the trait. Similarly, Z_1 and Z_2 are the covariates. We assume that Y_1 is available for all n subjects while G_1 is only available on $n_1(\in n)$ sequenced subjects. Since the covariates tend represent ancestry variables, it is reasonable to assume that Z_1 is only available for n_1 sequenced subjects. Next, regarding the secondary trait we assume that (Y_2, G_2, Z_2) are available on a further subset denoted n_2. Thus, the observed-data likelihood is

$$\prod_{i=1}^{n_1} P(Y_{1i}|G_{1i}, Z_{1i})P(G_{1i}, Z_{1i}) \prod_{i=n_1+1}^{n} \sum_{g,z} P(Y_1 i|g, z)P(g, z) \prod_{i=1}^{n_2} P(Y_{2i}|Y_{1i}, G_{2i}, Z_{2i}).$$

We can formulate the joint distribution of Y_1 and Y_2 through the bivariate linear regression model as:

$$Y_1 = \beta_1^T G_1 + \gamma_1^T Z_1 + \epsilon_1$$

$$Y_2 = \beta_2^T G_2 + \gamma_2^t Z_2 + \epsilon_2$$

where G_1 and G_2 are genotypes of individuals with Y_1 and Y_2 and $(\epsilon_1, \epsilon_2) \sim N_2(\mathbf{0}, \Sigma)$ $(\Sigma = \{\sigma_{kl} : k, l = 1, 2\})$. Therefore,

$$Y_2 = \delta \tilde{Y_1} + \beta_2^T G_2 + \gamma_2^T Z_2 + \tilde{\epsilon_2}$$

where $\delta = \sigma_{12}/\sigma_{11}, \tilde{Y_1} = Y_1 - \beta_1^T G_1 - \gamma_1^T Z_1$, and $\tilde{\epsilon_2} \sim N(0, \tilde{\sigma_{22}} = \sigma_{22} - \sigma_{12}^2/\sigma_{11})$.

2.1 Estimating parameters of primary trait

We maximize the first two terms in the observed-data likelihood function to obtain the MLEs of $(\beta_1, \gamma_1, \sigma_{11})$ and $P(.,.)$. A non-parametric MLE approach to estimate $P(.,.)$ is to ultilize discrete probabilities which put mass at distinct pairs the observed values $(g_1, z_1) \ldots (g_m, z_m)$ of (G_{1i}, Z_{1i}). We denote the point mass at (g_j, z_j) as q_j for $j = 1, \ldots, m$. The objective function which we maximize now becomes:

$$\sum_{i=1}^{n_1} \Big[\log P(Y_{1i}|G_{1i}, Z_{1i}) + \log \sum_{j=1}^{m} I\big\{(G_{1i}, Z_{1i}) = (g_j, z_j)\big\} q_j \Big] + \sum_{i=n_1+1}^{n} log \sum_{j=1}^{m} P(Y_{1i}|g_j, z_j) q_j.$$

Through EM algorithm, we estimates the parameters by iterating between the following E-step and M-step until convergence.

E-step. We have a n x m matrix ψ which is populated as followed. For $i = 1, \ldots, n_1$, $\psi_{ij} = I\{(G_{1i}, Z_{1i}) = (g_j, z_j)\}$ and for $i = n_i + 1, \ldots, n$, we set $\psi_{ij} = \frac{P(Y_{1i}|g_j, z_j) q_j}{\sum_{k=1}^{m} P(Y_{1i}|g_k, z_k) q_k}$, where $P(y_1|g, z) = (2\pi\sigma_{11})^{-1/2} \exp\Big(- \frac{(y_1 - \beta_1^T g - \gamma_1^T z)^2}{2\sigma_{11}} \Big)$.

M-Step. We update each parameter as followed:

- $\eta = \Big(\sum_{i=1}^{n} \sum_{j=1}^{m} \psi_{ij} W_j W_j^T \Big)^{-1} \Big(\sum_{i=1}^{n} Y_{1i} \sum_{j=1}^{m} \psi_{ij} W_j \Big)$, where $\eta = \begin{bmatrix} \beta_1 \\ \gamma_1 \end{bmatrix}, W_j = \begin{bmatrix} g_j \\ z_j \end{bmatrix}$.
- $\sigma_{11} = n^{-1} \sum_{i=1}^{n} \sum_{j=1}^{m} \psi_{ij} (Y_{1i} - \eta^T W_j)^2$.
- $q_j = n^{-1} \sum_{i=1}^{n} \psi_{ij}, j = 1, \ldots, m$.

Next, I will calculate the asymptotic covariance matrix. Let ℓ_{1ij} and ℓ_{2ij} be the first and second derivative of $\log P(Y_{1i}|g_j, z_j) + \log q_j$. The information marix is $Q_1 = -\sum_{i=1}^{n} \sum_{j=1}^{m} \psi_{ij} \ell_{2ij} - \sum_{i=1}^{n} \Big\{ \sum_{j=1}^{m} \psi_{ij} \ell_{1ij} \ell 1ij^T - (\sum_{j=1}^{m} \psi_{ij} \ell_{1ij})(\sum_{j=1}^{m} \psi_{ij} \ell_{1ij})^T \Big\}$. Since there is a constraint: $\sum_{j=1}^{m} q_j = 1$, let D be the derivative matrix of $(\beta_1, \gamma_1, \sigma_{11}, q_1, \ldots, q_m)$ with respect to $(\beta_1, \gamma_1, \sigma_{11}, q_1, \ldots, q_{m-1})$. Thus, the covariance matrix for these parameters is $\Omega_1 = F^{-1}$, where $F = D^T Q_1 D$.

2.2 Estimating parameters for secondary trait

We obtain the parameters of the secondary trait by maximizing the last term in the likelihood. However, this is equivalent to using ordinary least square method to the observations $(Y_{2i}, \hat{Y}_{1i}, G_{2i}, Z_{2i})$, where $i = 1, \ldots, n_2$ and $\hat{Y}_{1i} = Y_{1i} - \hat{\beta}_1^T G_{1i} - \hat{\gamma}_1^T Z_{1i}$. Thus, the estimates for $(\delta, \beta_2, \gamma_2)$ are $\begin{bmatrix} \hat{\delta} \\ \hat{\beta}_2 \\ \hat{\gamma}_2 \end{bmatrix} = \left(\sum_{i=1}^{n_2} \begin{bmatrix} \hat{Y}_{1i} \\ G_{2i} \\ Z_{2i} \end{bmatrix}^{\otimes 2} \right)^{-1} \left(\sum_{i=1}^{n_2} Y_{2i} \begin{bmatrix} \hat{Y}_{1i} \\ G_{2i} \\ Z_{2i} \end{bmatrix} \right)$. We can estimate σ_{22} as followed, $\hat{\tilde{\sigma}}_{22} = n_2^{-1} \sum_{i=1}^{n_2} (Y_{2i} - \hat{\delta} \hat{Y}_{1i} - \hat{\beta}_1^T G_{2i} - \hat{\gamma}_2^T Z_{2i})^2$. Lastly, the covariance matrix is $\Omega_2 = \hat{\tilde{\sigma}}_{22} \left(\sum_{i=1}^{n_2} \begin{bmatrix} \hat{Y}_{1i} \\ G_{2i} \\ Z_{2i} \end{bmatrix}^{\otimes 2} \right)^{-1} + J \tilde{\Omega}_1 J^T$, where J is the Jacobian matrix of $(\hat{\delta}, \hat{\beta}_2, \hat{\gamma}_2)$ w.r.t. $(\hat{\beta}_2, \hat{\gamma}_2)$ and $\tilde{\Omega}_1$ is the block of Ω_1 corresponding to (β_1, γ_1).

3 Implementation of algorithm

The first version of the algorithm was written with the assumption that there are n_1 unique pairs of (q_j,z_j)$. It was the most straightforward way to get a working algorithm. The profling and benchmark of the algorithm (refers to 3 - Algorithm for more details) reveals that the estimation of parameters for primary trait requires ~13 seconds per run. Approximately 6 of these seconds were spent in estimating the covariance matrix. On the other hand, the secondary trait does not require as much time. Due to the fact that ordinary least square is a well-known regression, with a load of a python library, we could estimate these parameters in very little time. It is concerning that the algorithm takes a lot of time during the first phase; therefore, we focused on improving the implementation in the estimation for primary trait. The strategies were to vectorize and possibly utilize cython to improve speed.

In the second version of this algorithm, vectorization greatly improves the speed of estimation for primary trait (see 4 - Code profiling for more details). The speed of the estimation decreases by almost 50% (~6.5 seconds). While most of the time is spent in estimating the covariance, this implies that other parts of the algorithm are quite efficient. We attempted to use cython or just-in-time compilation to improve the speed for covariance estimation. However, it was difficult to implement due to the complexity of the algorithm. Therefore, we couldn't improve the speed for this part of the algorithm.

4 Application to simulated data

We simulate a dataset that mimicks the NHLBI ESP study from the reference paper. For a cohort of $n = 5000$, the primary and secondary traits are generated from the equations

$$Y_1 = \beta_1 G + \gamma_1 Z + \epsilon_1 Y_2 = \beta_2 G + \gamma_2 Z + \epsilon_2.$$

G represents the number of minor allele observed at a SNP which follows a binomial distribution n and probability 0.04 which is the minor allele frequency. Z is a normally distributed covariate representing a principal component for ancestry or other type of genetically related variable. It has mean g for $G = g$ and unit variance. Lastly, ϵ_1 and ϵ_2 follows a bivariate normal distribution with covariance 0.38. After generating Y_1 and Y_2, we obtain (G, Z, Y_2) of the smallest 150 values of Y_1 and the largest 150 values of Y_2. Thus, $n_1 = n_2 = 300$.

We obtain the parameters estimates for the primary and secondary trait when apply the algorithm to the simulated dataset. Since we only focus on the estimates of these parameters instead of testing, we could not access the performance of these estimates. The estimates are reflected as followed:

```
In [29]: import pandas as pd
         import numpy as np
         res = pd.read_csv("analysis_output.txt",delim_whitespace=True,skipinitialspace=True)
         print res
```

```
--------  -------------
0    beta_1       0.121687
1   gamma_1    0.000476686
2  sigma_11        406.562
3  --------  -------------
4  --------     ----------
5     delta       0.378936
6    beta_2       0.201665
7   gamma_2     -0.0100374
8  sigma_22       0.785484
9  --------     ----------
```

5 Conclusion

Failure to account for the sampling technique in these quantitative trait studies tends to yield inflated type I error rate. Even though we do not conduct a simulation study to demonstrate this point, the reference has strong evidence in supporting the well-controlled type I error rate of this algorithm. In future work, we will continue to optimize this algorithm to improve its speed performance. The first stage of vectorization increases the speed by almost 50% as demonstrated above. However, the speed can be improve even more by utilizing cython or just-in-time compilation.