Ankush Bharadwaj

404912130

CM124/224: Machine Learning Applications in Genetics

Winter 2020

Final Project Report

The problem I am solving in this project is as follows: given masked genotype data for *i* individuals across *n* SNPs, determine the haplotypes for each individual across these SNPs. My approach involved developing functions to solve this problem. First, one set of functions will need to convert the masked genotype matrix (*n* x *i*) that has values from the set {*, 0, 1, 2} into an unmasked genotype matrix (*n* x *i*) that has values from the set {0, 1, 2}. Next, another set of functions will need to convert the unmasked genotype matrix (*n* x *i*) with values from the set {0, 1, 2} into an unmasked haplotype matrix (*n* x 2**i*) with values from the set {0, 1}. After breaking the problem into two smaller sub-problems, the sets of functions to approach the overall problem were implemented in Python 3.

First, to unmask the masked genotype matrix (*n* x *i*), I developed a function to iterate through the rows of the masked genotype matrix. Within each row, I first found the count of each value in the row, which resulted in a Python 3 object with the number of instances of each value, which should be from the set {*, 0, 1, 2}, where '*' denotes an unknown genotype. Knowing that each unknown genotype is homozygous, I understood that each '*' would be replaced by either a '0' or a '2', as a '0' indicates that the haplotypes for that individual are (0, 0) and a '2' indicates that the haplotypes for that individual are (1, 1). To decide which genotype to replace the '*' with, I found the total number of known genotypes across the *i* individuals at the row, or SNP, I am currently at, as well as the total number of '2' genotypes across the *i* individuals. Using these two values, I found the probability of the '2' genotype and called a random function that utilizes a binomial distribution to decide whether the '*' at a specific location should be replaced with a '0' or a '2'. After iterating through the entire masked genotype matrix with this algorithm, all of the '*' values should be replaced with either a '2' or '0', depending on the frequency of the '2' and '0' genotypes at that SNP across the known individuals. Therefore, at the end of this set of functions, I have unmasked the masked genotype matrix (*n* x *i*).

The next step is to determine the haplotypes from the unmasked genotype matrix (*n* x *i*), which would result in a haplotype matrix of (*n* x 2**i*), with values from the set {0, 1}. Before continuing, it is crucial that I make sure that the unmasking occurred properly, so I once again iterate through the unmasked genotype matrix and return a 'False' Boolean value if any '*' are encountered. After the unmasking is verified, the haplotype matrix is constructed. Once again, I iterate through the unmasked genotype matrix, and this time, I am going to be imputing values into an empty haplotype matrix with n rows and $2*i$ columns, given an unmasked genotype matrix of n rows and i columns, as each individual column of the unmasked genotype matrix will result in two columns for the haplotype matrix. While iterating through the unmasked genotype matrix, I know that any location with a '0' or '2' would result in a haplotype of (0, 0) or (1, 1), respectively. This means that if I am currently at k row and j column of the unmasked genotype matrix, if I encounter a '0' at this location, the kth row and $j, j+1$ columns of the haplotype matrix would have '0' values, and if I encounter a '2' at this location, kth row and $j, j+1$ columns of the haplotype matrix would have '1' values. The only remaining genotype to resolve is anywhere I encounter a '1', which means that the haplotypes for an individual with this genotype would either be (0, 1) or (1, 0). To determine which haplotype to impute for that individual at that SNP, I utilized a pure parsimony approach: knowing that most haplotypes in a population are similar to each other, if I encounter a '1' at position $[k, j]$ in the unmasked genotype matrix, I look for the nearest completed haplotype value – most likely an individual with a homozygous allele – in the haplotype matrix, and impute that value for the position $[k, j]$ in the haplotype matrix, and the other value at position $[k, j+1]$ in the haplotype matrix. For example, if the nearest completed haplotype value is '1', the value at position $[k, j]$ in the haplotype matrix would be a '1' and the value at position $[k, j+1]$ would be a '0'. Using this algorithm, the haplotype matrix for each SNP and each individual is filled and written into a text file. Therefore, once the function has completed a full iteration of the unmasked genotype matrix, I have a resulting haplotype matrix.