Genomics

Freie Universität Berlin, Institut für Informatik
Knut Reinert, Peter Robinson, Sebastian Bauer
Wintersemester 2012/2013

8. Übungsblatt vom 11. Dezember 2012
Diskussion im kommenden Jahr (to be determined)

Exercise 1.

In this exercise, you will be guided through the implementation of a simple version of the EM algorithm to estimate the allele frequencies of the blood groups. Recall the blood groups A,B,AB, and O. The allel O is recessive, so that the genotypes AO and AA both correspond to the phenotype (i.e., blood group) A. Similarly, the two genotypes BO and BB correspond to blood group B, and the single genotype OO corresponds to the blood group O, and the genotype AB corresponds to the blood group AB.

Therefore, if we have measurements of the phenotype (blood group), can we make an estimate for the underlying allele frequencies, given that we cannot observe if somebody with the blood group A has the genotype AO or AA? This is a typical application of EM type algorithms.

Let us describe the allele frequencies as

A	B	O
μ_A	μ_B	μ_O

clearly, $\sum \mu_i = 1$. Furthermore, assuming Hardy-Weinberg equilibrium[1], then we can calculate the genotype frequencies as

AA	AO	BB	BO	AB	OO
μ_A^2	$2\mu_A\mu_O$	μ_B^2	$2\mu_B\mu_O$	$2\mu_A\mu_B$	μ_O^2

We have now model parameters $\Theta = \{\mu_A, \mu_B, \mu_O\}$, and an unobservable (hidden) genotype variable (one of the six genotypes listed above). We have an observed phenotype (blood group) variable (A, B, AB, or O). If we know the frequency of the genotypes, then it is trivial to calculate the frequency of the phenotypes. For instance,

$$P(\text{Blood group} = A|\Theta) = \mu_A^2 + 2\mu_A\mu_O \tag{1}$$

In this context, the EM algorithm works as follows. Imagine we know the true allele frequences $\{\mu_A, \mu_B, \mu_O\}$. Then, it is easy to estimate the number of people with blood group A in a population of N individuals. The genotype AA has a population frequency of μ^2 and the genotype AO has the frequency $2\mu_A\mu_O$. Therefore, if we observe that n_A out of the N people in our sample have

[1] I am assuming knowledge of ABO, alleles vs. genotypes, and Hardy Weinberg from your Bachelor curriculum. Check Wikipedia or contact me if you do not understand this. Recall that the heteroyzgote frequency at a locus with two alleles of frequency p and $q = 1 - p$ is $2pq$ according to the HW equilibrium.

blood group A, then we **expect** the following count of people with genotype AA (where $n_{A/A}$ is the count of people with the genotype AA in our sample):

$$\#(\text{genotype} = AA) = n_{A/A} = n_A \times \frac{\mu^2}{\mu^2 + 2\mu_A\mu_O} \tag{2}$$

and similarly we **expect** the following count of people with genotype AO

$$\#(\text{genotype} = AO) = n_{A/O} = n_A \times \frac{2\mu_A\mu_O}{\mu^2 + 2\mu_A\mu_O} \tag{3}$$

We can now **maximize** the likelihood of our estimates of the allele frequencies given our estimates of the counts of individuals with each genotype. For instance,

$$\mu_A^{'} = \frac{2 * n_{A/A} + n_{A/0} + n_{A/B}}{2 * N} \tag{4}$$

Note the use of the factor 2 for $n_{A/A}$. People with this genotype have two A alleles, and similarly, a population of N people has a total of $2N$ alleles.

Note that to solve this problem, you will also need to find equations for $n_{B/B}$ and $n_{B/O}$.

The assignment

Let us assume we have a sample of people for whom we have measured the blood group as follows.

A	B	AB	O	Total
$n_A = 186$	$n_B = 38$	$n_{AB} = 13$	$n_O = 284$	521

We want to estimate the allele frequencies for A,B, and O using the EM algorithm. Complete the following two functions. That is, replace the question marks (???) with R code. Note that it is not necessary to calculate $n_{A/B}$ and $n_{O/O}$ explicitly because these values are constrained by the values for $n_{A/A}$, $n_{A/O}$, $n_{B/B}$, and $n_{B/O}$.

```
ABOexpect <- function(na,nb,mu.a,mu.b,mu.o) {
  naa <- ???
  nao <- ???
  nbb <- ???
  nbo <- ???
  N <- list(aa=naa,ao=nao,bb=nbb,bo=nbo)
}
```

This function performs the expectation step. Now complete the following function, which estimates μ_A, μ_B, and μ_O.

```
 ABOmaximize <- function(naa,nao,nbb,nbo,nab,no) {
    n <-???
    a <- ???
    b <- ???
    o <- ???
    L <- list(mu.a=a,mu.b=b,mu.o=o)
}
```

The following code is provided. It iteratively performs the E and the M steps until their is no further change in estimates of μ_A, μ_B, and μ_O.

```
#initial guess
mu.a <- 0.3
mu.b <- 0.2
mu.o <- 0.5

## Observed data (counts of people with the four blood groups)
na <- 186
nb <- 38
nab <- 13
no <- 284

## Set up iteration
iter <- 1
diff <- 1
tol <- 0.0001 ## tolerance

while (diff > tol) {

      E <- ABOexpect(na,nb,mu.a,mu.b,mu.o)

      M <- ABOmaximize(E$aa,E$ao,E$bb,E$bo,nab,no)

      diff <- abs(M$mu.a - mu.a) + abs(M$mu.b - mu.b) + abs(M$mu.o - mu.o)
      mu.a <- M$mu.a
      mu.b <- M$mu.b
      mu.o <- M$mu.o
      cat(sprintf("iter:%d, diff:%.2f\n",iter,diff))
      iter <- iter + 1
}

cat(sprintf("mu_a=%.4f, mu_b=%.4f, mu_o=%.4f\n",mu.a,mu.b,mu.o))
```

Exercise 1.

Write R code for the two functions for the ABO gene counting EM algorithm

fragile]

```
ABOexpect <- function(na,nb,pa,pb,po) {
  naa <- na*pa^2/(pa^2+2*pa*po)
  nao <- na*2*pa*po/ (pa^2 + 2*pa*po)
  nbb <- nb*pb^2 / (pb^2+2*pb*po)
  nbo <- nb*2*pb*po / (pb^2+2*pb*po)
  N <- list(aa=naa,ao=nao,bb=nbb,bo=nbo)
}
ABOmaximize <- function(naa,nao,nbb,nbo,nab,no) {
    n <- naa+nao+nbb+nbo++nab+no
    a <- (2*naa + nao + nab)/(2*n)
    b <- (2*nbb+nbo+nab)/(2*n)
```

```
    o <- 1-a-b
    L <- list(mu.a=a,mu.b=b,mu.o=o)
}
```

Exercise 2.

Use your code to estimate the A,B,O allele frequencies

fragile]

```
> source("genecounting.R")
iter:1, diff:0.43
iter:2, diff:0.04
iter:3, diff:0.00
iter:4, diff:0.00
iter:5, diff:0.00
mu_a=0.2136, mu_b=0.0501, mu_o=0.7363
```

Exercise 3.

Try using different initial guesses for μ_A, μ_B, and μ_O. Does this affect the performance of the algorithm?

fragile] Yes, but not as much as one might expect. For instance,

```
mu.a <- 0.9
mu.b <- 0.05
mu.o <- 0.05
```

leads to

```
> source("genecounting.R")
iter:1, diff:1.10
iter:2, diff:0.26
iter:3, diff:0.03
iter:4, diff:0.00
iter:5, diff:0.00
iter:6, diff:0.00
mu_a=0.2136, mu_b=0.0501, mu_o=0.7363
```

That is, it took one additional iteration to get the same solution.