Seth Temple
sdtemple@uw.edu

EM: ABO Blood Types

STAT 499
November 5, 2020

Here we introduce the expectation-maximization algorithm through an example estimating allele frequencies for ABO blood types. This note is intended as a reference for graduate students, but may be used to present general concepts to advanced undergraduates. Appended are slides by Professor Sharon Browning, a worked homework example of mine, and notes from Professor Yen-Chi Chen.

This example summarizes how to implement an EM algorithm. We want to estimate proportions for the complete, but unobserved genotype data using the observed phenotype data. Consider the log-likelihood for the complete model:

$$\begin{aligned} l(p_a, p_b, p_o|\text{genotypes}) = n_{aa} \cdot 2\log p_a + n_{bb} \cdot 2\log p_b + n_{oo} \cdot 2\log p_o \\ + n_{ao}\log(2p_a p_o) + n_{bo}\log(2p_b p_o) + n_{ab}\log(2p_a p_b). \end{aligned}$$

Notably, we don't know the genotype counts $n_{aa}, n_{ao}, n_{ab}, n_{bb}, n_{bo}$ and n_{oo}. Consider the log-likelihood for the incomplete model when we only have phenotype counts. This model suggests the conditional expectations of genotype counts given phenotype counts.

$$\begin{aligned} l(p_a, p_b, p_o|\text{phenotypes}) = n_a \cdot \log(p_a^2 + 2p_a p_o) + n_b \log(p_b^2 + 2p_b p_o) \\ + n_{ab}\log(2p_a p_b) + n_o \log(p_o^2) \end{aligned}$$

E-step. Find $\mathbb{E}[n_{aa}|n_a]$ where $n_{aa}|n_a$ follows a binomial probability model. Do this to compute the six conditional expectations. This E-step is computing the expectation of the genotypes given the observed phenotypes; that is, computing the expectation of the complete data parameters given the observed data.

M-step. Return to the log-likelihood for the complete model. Use partial derivatives and possibly constrained maximization techniques (Lagrange multipliers) to determine the arg max for the parameters (p_a, p_b, p_o). Plug in your conditional expectations from the E-step where appropriate.

Termination. Iterate until you reach convergence (defined by some stopping rule).

- Consider many initializations of (p_a, p_b, p_o) to explore the parameter space.
- This procedure works well for exponential families (normal, binomial, gamma, etc.) because we know how to compute their conditional expectations and maximize their log-likelihoods.
- This procedure also works well for mixture models of exponential families, which are also exponential families.

ABO allele frequencies

For the ABO blood group the EM-algorithm is one of the easiest ways to find the MLEs of the allele frequencies.

Write p, q, r for the frequencies of the A, B and O alleles.

$$p + q + r = 1$$

We'll assume HWE. The frequencies of the genotypes AA, AB, AO, BB, BO, OO are $p^2, 2pq, 2pr, q^2, 2qr, r^2$.

Then the frequencies of the blood types A, B, AB, O are $p^2 + 2pr, q^2 + 2qr, 2pq, r^2$

ABO likelihood

- The counts n_A, n_B, n_{AB}, n_O for the 4 blood types are drawn from a Multinomial distribution (with probabilities on the last slide).
- The likelihood is

$$L(p,q,r) = (p^2 + 2pr)^{n_A}(q^2 + 2qr)^{n_B}(2pq)^{n_{AB}}(r^2)^{n_O}$$

- The log likelihood is $l(p,q,r) = n_A \log(p^2 + 2pr) + n_B \log(q^2 + 2qr) + n_{AB} \log(2pq) + n_O \log(r^2)$
- When we apply the EM algorithm, the likelihood (and log likelihood) are guaranteed to increase (or at least not decrease) from one iteration to the next.

E-step: partition the A phenotypes into expected counts of AA and AO genotypes, and similarly B into BB and BO:

$$\Pr(AO \mid \text{type } A) = 2pr/(p^2 + 2pr) = 2r/(p + 2r)$$
$$\Pr(BO \mid \text{type } B) = 2qr/(q^2 + 2qr) = 2r/(q + 2r)$$

M-step: Then $\tilde{p} = (2\Pr(AA) + \Pr(AO) + \Pr(AB))/2$

and $\tilde{q} = (2\Pr(BB) + \Pr(BO) + \Pr(AB))/2$

EM iterations for the ABO example

On a sample of 502 individuals, 42.2% are type A, 20.6% type B, 7.8% type AB, 29.4% type O.

With the starting values of $p = 0.3, q = 0.3$, the log likelihood is -687.12

p	q	$\frac{2r}{p+2r}$	$\frac{2r}{q+2r}$	Pr(AA)	Pr(AO)	Pr(BB)	Pr(BO)	Pr(AB)	P(OO)	$\tilde{p}$	$\tilde{q}$	$\tilde{l}$
0.300	0.300	0.73	0.73	0.115	0.307	0.056	0.150	0.078	0.294	0.308	0.170	-629.00
0.308	0.170	0.77	0.86	0.096	0.326	0.029	0.177	0.078	0.294	0.298	0.156	-627.57
0.298	0.156	0.79	0.87	0.091	0.331	0.026	0.180	0.078	0.294	0.295	0.155	-627.53
0.295	0.155	0.79	0.88	0.089	0.333	0.025	0.181	0.078	0.294	0.295	0.155	-627.52

Notice the log likelihood l always increases.

Bigger picture

- EM and EM-like algorithms a lot in genetics.
- Often in genetics we can model underlying processes, but we don’t get to observe all the data we’d like to see.
- Here the unobserved data were allele counts, in other settings they might be
  - The locations of recombination points.
  - The alleles that are on the same haplotype at two genetic markers.
  - The pattern of inheritance at a locus (and across loci) within a family.

Alternatives to the EM algorithm

- The EM algorithm tends to have slow convergence, although initial convergence is generally fast.
- As with all iterative algorithms, if the likelihood surface is not convex, the algorithm can find local maxima.
  - More likely to be a problem the more parameters there are.
  - Can try different starting points and pick the final solution with the highest likelihood.
- Sometimes other iterative algorithms for maximizing the likelihood have better convergence.
  - For example, the Newton-Raphson algorithm.

3. (a)

```
rm(list = ls())
ll <- function(p, n)
{
  a <- (n[1] + n[2]) * log(p[1] ^ 2 + 2 * p[1] * p[3])
  b <- (n[4] + n[5]) * log(p[2] ^ 2 + 2 * p[2] * p[3])
  o <- n[6] * log(p[3] ^ 2)
  ab <- n[3] * log(2 * p[1] * p[2])
  return(a + b + o + ab)
}

abo_pupdate <- function(p, n)
{
  p[1] <- (2 * n[1] + n[2] + n[3]) / (2 * sum(n))
  p[2] <- (2 * n[4] + n[5] + n[3]) / (2 * sum(n))
  p[3] <- (2 * n[6] + n[2] + n[5]) / (2 * sum(n))
  p[4] <- (2 * p[1] * p[3]) / (p[1] ^ 2 + 2 * p[1] * p[3])
  p[5] <- (2 * p[2] * p[3]) / (p[2] ^ 2 + 2 * p[2] * p[3])
  return(p)
}

abo_nupdate <- function(p, n)
{
  na <- n[1] + n[2]
  nb <- n[4] + n[5]
  n[1] <- na * (1 - p[4])
  n[2] <- na * p[4]
  n[4] <- nb * (1 - p[5])
  n[5] <- nb * p[5]
  return(n)
}

# initialize
p <- rep(1 / 3, 3)
na <- 47; nb <- 38; nab <- 8; no <- 54
p <- c(p, (2 * p[1] * p[3]) / (p[1] ^ 2 + 2 * p[1] * p[3]))
p <- c(p, (2 * p[2] * p[3]) / (p[2] ^ 2 + 2 * p[2] * p[3]))

# zeroeth step
nao <- na * p[4]; naa <- na * (1 - p[4])
nbo <- nb * p[5]; nbb <- nb * (1 - p[5])
n <- c(naa, nao, nab, nbb, nbo, no); rm(naa, nao, nbb, nbo)
ll0 <- -Inf; ll1 <- ll(p, n)

# loop until convergence
ct <- 0
while(ll1 > ll0){
  ll0 <- ll1
  n <- abo_nupdate(p, n)
  p <- abo_pupdate(p, n)
  ll1 <- ll(p, n)
  ct <- ct + 1
}
```

```
print(paste("convergence after", ct, "steps"))
```

```
## [1] "convergence after 13 steps"
```

```
print(paste('Frequency estimates for A, B, and O are',
            round(p[1], 2), round(p[2], 2), round(p[3], 2)))
```

```
## [1] "Frequency estimates for A, B, and O are 0.21 0.17 0.62"
```

(b)

```
print(paste("Log-likelihood for EM esimates is",
            round(ll0, 4)))
```

```
## [1] "Log-likelihood for EM esimates is -182.9029"
```

```
p0 <- c(.25, .25, .50)
p0 <- c(p0, (p0[1] ^ 2) / (p0[1] ^ 2 + 2 * p0[1] * p0[3]))
p0 <- c(p0, (p0[2] ^ 2) / (p0[2] ^ 2 + 2 * p0[2] * p0[3]))
naa0 <- na * p0[4]; nao0 <- na * (1 - p0[4])
nbb0 <- nb * p0[5]; nbo0 <- nb * (1 - p0[5])
n0 <- c(naa0, nao0, nab, nbb0, nbo0, no);
rm(naa0, nbb0, nao0, nbo0)
print(paste("Log-likelihood for provided frequencies is",
            round(ll(p0, n0), 4)))
```

```
## [1] "Log-likelihood for provided frequencies is -190.3632"
```

STAT 516: Stochastic Modeling of Scientific Data **Autumn 2019**

Lecture 8: EM Algorithm and Gradient Descent

Instructor: Yen-Chi Chen

These notes are partially based on those of Mathias Drton.

8.1 Introduction

In statistics, often we focus a lot on how to design a good estimator with nice theoretical properties (such as consistency, convergence rate, good posterior distribution). However, this is often not enough when we are working with a realistic dataset. When analyzing the data, we need to be able to *numerically compute* the estimator. In many cases, we can propose an estimator that has many good theoretical properties such as uniqueness, fast convergence rate, asymptotic normality, but we are not able to compute it.

For one example, consider a simple linear regression $Y = X^T\beta + \epsilon$, where we observe $n = 1000$ data points and the covariates $X \in \mathbb{R}^{100}$. Assume that half of the parameters β are 0 (i.e., 50 entries of β are 0). Our goal is to find the the ones that are not 0. You can easily see that we can try all combination of β's (there will be totally $\binom{100}{50}$ combinations) and choose the one that best fit to the data. Under suitable assumptions, the estimator is unique, consistent, and has asymptotic normality. However, can we really find all combinations? Mathematically, yes–there are only $\binom{100}{50}$. But practically, very difficult! There are about 10^{29} combinations!

In this lecture, we will talk about numerical methods that help us to find statistical estimators. We will start with the EM algorithm, a famous method in statistics for finding a maximizer, and then talk about a more general approach called gradient descent/ascent method along with a stochastic version of it.

Before we proceed, we start with a simple example about finding the ratio of certain gene. Suppose we collect n individuals from a population and for each individual, his/her blood type $f_i \in \{A, B, AB, O\}$. Recall that blood types are determined by six possible genotypes $g_i \in \{AA, AO, BB, BO, AB, OO\}$ that corresponds to

$$\begin{aligned} g_i = AA \text{ or } AO &\Rightarrow f_i = A \\ g_i = BB \text{ or } BO &\Rightarrow f_i = B \\ g_i = AB &\Rightarrow f_i = AB \\ g_i = OO &\Rightarrow f_i = O. \end{aligned}$$

The data we have is $\{f_1, \cdots, f_n\}$ and our goal is to estimate the ratio of genes A, B, and O (we will denote them by p_A, p_B, p_O). Note that in the language of missing data, $G_n = \{g_1, \cdots, g_n\}$ will be called the complete-data and $F_n = \{f_1, \cdots, f_n\}$ will be called the observed-data. Often G_n are unobserved and we only have access to the observed data F_n so the goal is to see if we can recover p_A, p_B, p_O with the observed data.

Under the Hardy-Weinberg equilibrium, the number of each genotype is drawn from a multinomial distribution with

$$\begin{aligned} P(g_i = AA) = p_A^2, P(g_i = AO) = 2p_Ap_O, P(g_i = BB) = p_B^2, \\ P(g_i = BO) = 2p_Bp_O, P(g_i = AB) = 2p_Ap_B, P(g_i = OO) = p_O^2. \end{aligned}$$

Thus, if we observed the complete-data, the likelihood function is

$$L(p_A, p_B, p_O|G_n) \propto (p_A^2)^{m_{AA}}(2p_Ap_O)^{m_{AO}}(p_B^2)^{m_{BB}}(2p_Bp_O)^{m_{BO}}(2p_Ap_B)^{m_{AB}}(p_O^2)^{m_{OO}},$$

where each $m_{KL} = \sum_{i=1}^{n} I(g_i = KL)$. This likelihood function is often called the *complete data likelihood*. Finding the MLE of (p_A, p_B, p_O) is not very difficult in this case.

However, in reality, we do not observe G_n but instead, we only have F_n. So the likelihood function we are actually working with is

$$L(p_A, p_B, p_O|F_n) \propto (p_A^2 + 2p_A p_O)^{n_A} (p_B^2 + 2p_B p_O)^{n_B} (2p_A p_B)^{n_{AB}} (p_O^2)^{n_O}, \tag{8.1}$$

where $n_K = \sum_{i=1}^{b} I(f_i = K)$ is the number of blood type K. There is no closed-form of the MLE of equation (8.1). Thus, we need to use some numerical methods to find it.

8.2 EM Algorithm

The EM (Expectation Maximization) algorithm offers a simple and elegant way to finding an MLE when the likelihood function is complex. The EM is often applied to the case where the model involves *hidden/latent units*. Latent variable models and missing data are two common scenarios that it can be applied to.

Here we describe the general formulation of the EM algorithm in simple missing data problem. Let $\mathbf{x}$ be the complete data and $\mathbf{y}$ be the observed data and let $L(\theta|\mathbf{x}) = p(\mathbf{x};\theta)$ be the likelihood function (on the complete data). Given an initial guessThe EM algorithm keeps iterates the following two steps:

- E-step: evaluate $Q(\theta;\theta^{(n)}|\mathbf{y}) = \mathbb{E}(\log L(\theta|\mathbf{x})|\mathbf{y};\theta^{(n)})$,
- M-step: update $\theta^{(n+1)} = \mathsf{argmax}_\theta Q(\theta;\theta^{(n)}|\mathbf{y})$,

until certain criterion is met (e.g., $\|\theta_{n-1} - \theta^{(n)}\|_\infty < \epsilon$). Note that $\mathbb{E}(\log L(\theta|\mathbf{x})|\mathbf{y};\theta^{(n)})$ means that $\mathbf{x}$ is from the distribution $p(\cdot;\theta^{(n)})$ conditional on $\mathbf{y}$.

The EM algorithm has a powerful property about *ascending likelihood*.

Proposition 8.1 (Ascending property of EM)

$$\ell(\theta^{(n+1)}|\mathbf{y}) = \log p(\mathbf{y};\theta^{(n+1)}) \geq \log p(\mathbf{y};\theta^{(n)}) = \ell(\theta^{(n)}|\mathbf{y}).$$

Namely, the likelihood value will not decrease after each step of EM.

Although Proposition 8.1 states that the likelihood value is non-decreasing after each iteration, it does not guarantee that it is always increasing and also, it does not guarantee to find the global maximizer (MLE).

Proof:

The key step of the proof is the Jensen's inequality, which regularizes the expectation of a convex function. A function $f : \mathbb{R}^d \mapsto \mathbb{R}$ is called *convex* if its domain is a convex set and for any $\alpha \in [0,1]$ and any $a, b \in \mathbb{R}^d$,

$$\alpha f(a) + (1-\alpha) f(b) \geq f(\alpha a + (1-\alpha) b).$$

Jensen's inequality: if g is a convex function, then

$$\mathbb{E}(f(X)) \geq f(\mathbb{E}(X))$$

for any random variable X. If the function f is concave, then

$$\mathbb{E}(f(X)) \leq f(\mathbb{E}(X))$$

Since $\mathbf{x}$ is the complete-data and $\mathbf{y}$ is the observed-data,

$$p(\mathbf{x};\theta) = p(\mathbf{x},\mathbf{y};\theta) = p(\mathbf{x}|\mathbf{y};\theta)p(\mathbf{y};\theta).$$

Therefore,

$$\ell(\theta|\mathbf{y}) = \log p(\mathbf{y};\theta) = \log p(\mathbf{x};\theta) - \log p(\mathbf{x}|\mathbf{y};\theta).$$

Recall that we want to compute the difference of likelihood function under $\theta^{(n+1)}$ versus $\theta^{(n)}$, which is

$$\ell(\theta^{(n+1)}|\mathbf{y}) - \ell(\theta^{(n)}|\mathbf{y}) = \log p(\mathbf{x};\theta^{(n+1)}) - \log p(\mathbf{x};\theta^{(n)}) - \left\{\log p(\mathbf{x}|\mathbf{y};\theta^{(n+1)}) - \log p(\mathbf{x}|\mathbf{y};\theta^{(n)})\right\}.$$

Since $\theta^{(n+1)} = \mathsf{argmax}_\theta Q(\theta|\mathbf{y};\theta^{(n)})$, we want to associate this with the function Q. Thus, we take expectation of both sides of the random variable $\mathbf{x}$ conditional on $\mathbf{y}$ and from the distribution $p(\cdot;\theta^{(n)})$, leading to

$$\begin{aligned}
\ell(\theta^{(n+1)}|\mathbf{y}) - \ell(\theta^{(n)}|\mathbf{y}) &= \mathbb{E}\left\{\log p(\mathbf{x};\theta^{(n+1)}) - \log p(\mathbf{x};\theta^{(n)})|\mathbf{y};\theta^{(n)}\right\} \\
&\quad - \mathbb{E}\left\{\log p(\mathbf{x}|\mathbf{y};\theta^{(n+1)}) - \log p(\mathbf{x}|\mathbf{y};\theta^{(n)})|\mathbf{y};\theta^{(n)}\right\} \\
&= \underbrace{Q(\theta^{(n+1)}|\mathbf{y};\theta^{(n)}) - Q(\theta^{(n)}|\mathbf{y};\theta^{(n)})}_{\geq 0} - \mathbb{E}\left\{\log\left[\frac{p(\mathbf{x}|\mathbf{y};\theta^{(n+1)})}{p(\mathbf{x}|\mathbf{y};\theta^{(n)})}\right]|\mathbf{y};\theta^{(n)}\right\} \\
&\geq -\mathbb{E}\left\{\log\left[\frac{p(\mathbf{x}|\mathbf{y};\theta^{(n+1)})}{p(\mathbf{x}|\mathbf{y};\theta^{(n)})}\right]|\mathbf{y};\theta^{(n)}\right\}.
\end{aligned}$$

Finally, we apply the Jensen's inequality to the last quantity using the fact that log is a concave function, which implies

$$\begin{aligned}
\mathbb{E}\left\{\log\left[\frac{p(\mathbf{x}|\mathbf{y};\theta^{(n+1)})}{p(\mathbf{x}|\mathbf{y};\theta^{(n)})}\right]|\mathbf{y};\theta^{(n)}\right\} &\leq \log \mathbb{E}\left\{\frac{p(\mathbf{x}|\mathbf{y};\theta^{(n+1)})}{p(\mathbf{x}|\mathbf{y};\theta^{(n)})}|\mathbf{y};\theta^{(n)}\right\} \\
&\quad \log \frac{p(\mathbf{x}|\mathbf{y};\theta^{(n+1)})}{p(\mathbf{x}|\mathbf{y};\theta^{(n)})} p(\mathbf{x}|\mathbf{y};\theta^{(n)})d\mathbf{x} \\
&\quad = \log(1) = 0.
\end{aligned}$$

Thus, we have shown that

$$\ell(\theta^{(n+1)}|\mathbf{y}) - \ell(\theta^{(n)}|\mathbf{y}) \geq -\mathbb{E}\left\{\log\left[\frac{p(\mathbf{x}|\mathbf{y};\theta^{(n+1)})}{p(\mathbf{x}|\mathbf{y};\theta^{(n)})}\right]|\mathbf{y};\theta^{(n)}\right\} \geq 0,$$

which completes the proof.

■

Now we return to our blood type problem. We will illustrate how do we do EM algorithm in this case. To determine the E-step, we need the complete data likelihood function, which is

$$L(p_A, p_B, p_O|G_n) \propto (p_A^2)^{m_{AA}}(2p_Ap_O)^{m_{AO}}(p_B^2)^{m_{BB}}(2p_Bp_O)^{m_{BO}}(2p_Ap_B)^{m_{AB}}(p_O^2)^{m_{OO}}.$$

The log-likelihood function is

$$\begin{aligned}
\ell(p_A, p_B, p_O|G_n) &= m_{AA}\log(p_A^2) + m_{AO}\log(2p_Ap_O) + m_{BB}\log(p_B^2) \\
&\quad + m_{BO}\log(2p_Bp_O) + m_{AB}\log(2p_Ap_B) + m_{OO}\log(p_O^2) + C_0,
\end{aligned}$$

where C_0 is some constant independent of p.

E-step. Let $p = (p_A, p_B, p_O)$. The Q function under t-th iteration is then

$$\begin{aligned}
Q(p|F_n; p^{(t)}) &= \mathbb{E}(\ell(p|G_n)|F_n; p^{(t)}) \\
&= \mathbb{E}\{m_{AA}|F_n; p^{(t)}\} 2\log p_A + \mathbb{E}\{m_{AO}|F_n; p^{(t)}\}\log(p_A p_O) + \mathbb{E}\{m_{BB}|F_n; p^{(t)}\} 2\log p_B \\
&\quad + \mathbb{E}\{m_{BO}|F_n; p^{(t)}\}\log(p_B p_O) + \mathbb{E}\{m_{AB}|F_n; p^{(t)}\}\log(p_A p_B) + \mathbb{E}\{m_{OO}|F_n; p^{(t)}\} 2\log p_O + C_0.
\end{aligned}$$

Because $n_A = \sum_{i=1}^n I(f_i = A) = \sum_{i=1}^n I(g_i = AA \text{ or } AO)$. So

$$m_{AA}|n_A; p^{(t)} \sim \mathsf{Bin}\left(n_A, \frac{p_A^{(t)2}}{p_A^{(t)2} + 2p_A^{(t)}p_O^{(t)}}\right).$$

Thus,

$$\mathbb{E}\{m_{AA}|F_n; p^{(t)}\} = n_A \frac{p_A^{(t)2}}{p_A^{(t)2} + 2p_A^{(t)}p_O^{(t)}}.$$

The other conditional expectations can be derived in a similar way:

$$\begin{aligned}
m_{AA}^{(t)} &= \mathbb{E}\{m_{AA}|F_n; p^{(t)}\} = n_A \frac{p_A^{(t)2}}{p_A^{(t)2} + 2p_A^{(t)}p_O^{(t)}} \\
m_{AO}^{(t)} &= \mathbb{E}\{m_{AO}|F_n; p^{(t)}\} = n_A \frac{2p_A^{(t)}p_O^{(t)}}{p_A^{(t)2} + 2p_A^{(t)}p_O^{(t)}} \\
m_{BB}^{(t)} &= \mathbb{E}\{m_{BB}|F_n; p^{(t)}\} = n_B \frac{p_B^{(t)2}}{p_B^{(t)2} + 2p_B^{(t)}p_O^{(t)}} \\
m_{BO}^{(t)} &= \mathbb{E}\{m_{BO}|F_n; p^{(t)}\} = n_B \frac{2p_B^{(t)}p_O^{(t)}}{p_B^{(t)2} + 2p_B^{(t)}p_O^{(t)}} \\
m_{AB}^{(t)} &= \mathbb{E}\{m_{AB}|F_n; p^{(t)}\} = n_{AB} \\
m_{OO}^{(t)} &= \mathbb{E}\{m_{OO}|F_n; p^{(t)}\} = n_{OO}.
\end{aligned}$$

Using this, we can rewrite the Q function as

$$Q(p|F_n; p^{(t)}) = (2m_{AA}^{(t)} + m_{AO}^{(t)} + m_{AB}^{(t)})\log p_A + (2m_{BA}^{(t)} + m_{BO}^{(t)} + m_{AB}^{(t)})\log p_B + (2m_{OO}^{(t)} + m_{AO}^{(t)} + m_{BO}^{(t)})\log p_O.$$

M-step. Based on the Q function, the M-step is the traditional maximization problem with a constraint $p_A + p_B + p_O = 1$. We can use Lagrangian multiplier to solve it, which gives

$$\begin{aligned}
p_A^{(t+1)} &= \frac{2m_{AA}^{(t)} + m_{AO}^{(t)} + m_{AB}^{(t)}}{2n} \\
p_B^{(t+1)} &= \frac{2m_{BB}^{(t)} + m_{BO}^{(t)} + m_{AB}^{(t)}}{2n} \\
p_O^{(t+1)} &= \frac{2m_{OO}^{(t)} + m_{AO}^{(t)} + m_{BO}^{(t)}}{2n}
\end{aligned}$$

Starting from an initial guess $p^{(0)}$, we then iterates the E-step and M-step to obtain

$$p^{(1)}, p^{(2)}, \cdots$$

until certain stopping rule is satisfied and we use the final parameter value as our *computed MLE*.

In the above ideal case, both E-step and M-step has a closed form. However, in general they may not have a closed form so we need to use other numerical approach to approximate it. The E-step can be approximated by sampling the complete data $\mathbf{x}$ from the conditional PDF/PMF $p(\cdot|\mathbf{y};\theta^{(t)})$ and then use the average as a Monte Carlo estimator of the Q function. This idea is called the *Monte Carlo* EM algorithm[1]. The M-step may not have a closed-form as well. In this case, we need to use numerical optimization method, such as the gradient ascent method (we will discuss it later).

Note that only certain MLEs can be found by the EM algorithm; not all MLE can be found using the EM. Thus many research is about how to design an EM algorithm for finding MLE under certain problems.

Remark.

- A notable issue of the EM-algorithm (and other numerical methods as well) is the critical points problem. Although EM algorithm will not decrease the likelihood value, it may stuck at a critical point (local maxima or saddle points). If the likelihood function is convex, often this will not be a big problem but if the likelihood function is non-convex (which, unfortunately, is often the case), this could be a serious issue. Thus, often people will reinitialize the starting point of the EM multiple times and choose the convergent point that has the highest likelihood value. However, even doing so we may still not getting the true MLE.
- In the theoretical analysis of the EM-algorithm, people often focus on the Q function:

  $$Q(\theta;\theta^{(n)}|\mathbf{y}) = \mathbb{E}(\log L(\theta|\mathbf{x})|\mathbf{y};\theta^{(n)})$$

  because essentially, the EM is to update θ by maximizing $Q(\theta;\theta^{(n)}|\mathbf{y})$. This Q function is a sample quantity and it has a population version of it

  $$Q(\theta;\theta') = \mathbb{E}\{Q(\theta;\theta'|\mathbf{y})\}.$$

  If we update $\theta^{(n)}$ according to $Q(\theta;\theta^{(n)})$, this is called the *population EM algorithm*. The smoothness of this function and the behavior of $Q(\theta;\theta_{MLE})$ play key role in the convergence of EM[2].
- As you may notice, in the proof of likelihood ascent property, we only need $Q(\theta^{(n+1)}|\mathbf{y};\theta^{(n)}) - Q(\theta^{(n)}|\mathbf{y};\theta^{(n)})$. Thus, even if we do not maximization in the M-step, the likelihood value will be non-decreasing as long as $\theta^{(n+1)}$ has a higher value in the Q function. This idea was further developed into the EM gradient algorithm[3], which only finds a new $\theta^{(n+1)}$ that has a higher Q function value. This is particularly useful when the maximization is intractable. The paper of Kenneth Lange (1995) provides examples where the maximization is hard to compute so EM cannot be applied but the EM gradient works nicely.

8.3 Gradient Descent/Ascent

Gradient descent algorithm is a generic approach of finding a minimum of a smooth function. Note that finding the minimum and finding the maximum are almost equivalent question (just flip f to $-f$ or $1/f$) so here we will focus on finding the minimum. Let f be a smooth function. Out goal is to find $x^* = \mathsf{argmin}_x f(x)$, the location where the minimum of f occurs, and the minimal value $f^* = \min_x f(x)$.

[1] Please see "*implementations of the Monte Carlo EM Algorithm*" by Richard A. Levine and George Casella and https://arxiv.org/abs/1206.4768 for more details.

[2] If you are interested in, I would recommend this paper: https://arxiv.org/abs/1408.2156

[3] "A Gradient Algorithm Locally Equivalent to the EM Algorithm" by Kenneth Lange (1995).