1 Intrduction to the EM Algorithm

The EM (Expectation-Maximization) Algorithm is a general optimization method that is often applied to find the maximum likelihood estimates when missing values are present in a data set. The algorithm arose from the seminal paper, *Maximum Likelihood for incomplete data via the EM algorithm* by Dempster, Laird, and Rubin, and has since been widely applied and extended to other types of statistical problems. In particular, the algorithm is known to be applicable in a range of Bayesian estimation problems.

Incompleteness of data may arise from missing data, and this is a typical scenario when dealing with multivariate samples, or when dealing with censored data, or perhaps when latent variables are involved. Latent variables are unobserved variables which may be introduced to simplify the analysis in some way.

The main idea of the EM algorithm is relatively simple, and although it may be slow to converge when it is compared to other optimization methods, it is known that the EM algorithm is very reliable when searching for the global maximum. To begin, we start with an inital estimate of the target parameter, and then we alternate the E (expectation) step and M (maximization) step. In the E step, compute the conditional expectation of the objection function. The objective function is typically a log-likelihood function given the observed data and current parameter estimates. In the M step, the conditional expectation is maximized with respect to the target parameter. We update the estimates and iteratively repeat the E step and M step until the algorithm converges according to some criterion. Examples of convergence criterion included a fixed number of steps, or when the target parameter estimates fail to continue to vary more than a fixed value. Although the main idea is theoretically simple, for some problems, computing the conditional expectation in the E step can be troublesome. For incomplete data, the E step requires computing the the conditional expectation of a function of the complete data, given the missing data.

2 Example: EM Algorithm for a Mixture Model

I'd like to give an example of the EM algorithm at work. One may find that examples of the EM algorithm vary widely, so perhaps the best way to understand the procedure is to simply look at a good number of different examples. In this example, the EM algorithm is applied to estimate the parameters of the quadratic form

$$Y = \lambda_1 X_1^2 + \ldots \lambda_k X_k^2$$

where the X_i are i.i.d. standard Normal random variables, and $\lambda_1 > \lambda_2 > \ldots \lambda_k > 0$. One can apply elementary transformations to see that each $Y_i = \lambda_i X_i^2$ has a Gamma distribution with rate parameter $1/(2\lambda_i)$ and shape parameter $1/2$. Hence Y can be represented as the mixture of k independent Gamma random variables:

$$Y \stackrel{d}{=} \frac{1}{k} G(\frac{1}{2}, \frac{1}{2\lambda_1}) + \ldots + \frac{1}{k} G(\frac{1}{2}, \frac{1}{2\lambda_k})$$

Suppose now that $k = 3$ and that $\sum_{i=1}^{k} \lambda_i = 1$. Then the model has two unknown parameters as one can write

$$Y = \lambda_1 X_1^2 + \lambda_2 X_2^2 + (1 - (\lambda_1 + \lambda_2)) X_3^2$$

The EM algorithm first updates the posterior probability p_{ij} that the i^{th} sample observation y_i was generated from the j^{th} component. At the t^{th} step,

$$p_{ij}^{(t)} = \frac{\frac{1}{k} f_j(y_i | y, \lambda^{(t)})}{\sum_{j=1}^{k} \frac{1}{k} f_j(y_j | y, \lambda^{(t)})},$$

where $\lambda^{(t)}$ is the current estimate of the parameters $\{\lambda_j\}$, and $f_j(y_i | y, \lambda^{(t)})$ is the $Gamma(1/2, 1/(2\lambda_j^{(t)}))$ density evaluated at y_i. Note that the mean of the j^{th} component is λ_j, so the updating equation is

$$\mu_j^{(t+1)} = \frac{\sum_{i=1}^{m} p_{ij}^{(t)} y_i}{\sum p_{ij}^{(t)}}.$$

What follows is basic `R` script that performs an implementation of this algorithm for us:

```
m <-2000
> lambda <- c(.6 , .25, .15) #rate is given by 1/(2lambda)
> lam <- sample(lambda , size = 2000, replace=TRUE)
> y <- rgamma(m, shape = .5, rate = 1/(2*lam))
>
> N <- 10000 #maximum number of iterations
> L <- c(.5, .4, .1) #initial target estimates
> tol <- 0.000000001
> L.old <- L+1
>
> for (j in 1:N){
+     f1 <- dgamma(y , shape = 0.5, rate = 1/(2*L[1]))
+     f2 <- dgamma(y , shape = 0.5, rate = 1/(2*L[2]))
+     f3 <- dgamma(y , shape = 0.5, rate = 1/(2*L[3]))
+     py <- f1 / (f1 + f2 + f3) #posterior probability y from 1
```

```
+     qy <- f2 / (f1 + f2 + f3)
+     ry <- f3 / (f1 + f2 + f3)
+
+     mu1 <- sum(y*py) / sum(py) #update means
+     mu2 <- sum(y*qy) / sum(qy)
+     mu3 <- sum(y*ry) / sum(ry)
+     L <- c(mu1, mu2, mu3) #update lambdas
+     L <- L / sum(L)
+
+     if ( sum( abs( L - L.old) / L.old) < tol) break
+     L.old <-L
+
+ }
>
> print(list(lambda = L/sum(L) , iter =j, tol =tol))
$lambda
[1] 0.5350399 0.3077124 0.1572477

$iter
[1] 623

$tol
[1] 1e-09

>
```

Here the EM algorithm converged in 623 iterations (within $< 1e-9$) to the estimate $\hat{\lambda} \doteq (0.535, 0.308, 0.157)$. The data was generated with the parameters $(0.60, 0.25, 0.15)$.