Gaussian EM Algorithm

Greg Johnson

The following is an implementation of the classic EM-algorithm for missing multivariate Gaussian data.

```
X <- matrix(c(NA, 4.605047, 5.8303953, 7.595643, 1.754275, 1.8826819,
    4.047683, -1.791576, NA, -1.672295, -3.434457, 2.1768536,
    2.904052, -3.906055, -4.6161726), 5, byrow = TRUE)

EM_normal <- function(data, mu0, E0, tol = 1e-05) {

    muhat <- mu0
    Ehat <- E0
    X <- data
    n <- nrow(X)
    p <- ncol(X)
    mre <- 10000  #modified relative error - something large/arbitrary to enter the while loop
    count <- 0
    while (mre > tol) {
        count <- count + 1
        # Step one: Predict
        T1 <- 0
        T2 <- 0
        for (j in 1:n) {
            if (sum(is.na(X[j, ])) == 0) {
                x <- X[j, ]
                T1 <- T1 + x
                T2 <- T2 + x %*% t(x)
                next
            } else {
                mis <- is.na(X[j, ])
                x1 <- X[j, mis]  #missing components
                x2 <- X[j, !mis]  #not missing
                # we estimate x1 with the conditional expectation of x1 given
                # x2
                muhat1 <- muhat[mis]
                muhat2 <- muhat[!mis]
                Ehat11 <- Ehat[mis, mis]
                Ehat12 <- Ehat[mis, !mis]
                Ehat21 <- Ehat[!mis, mis]
                Ehat22 <- Ehat[!mis, !mis]
                x1est <- muhat1 + Ehat12 %*% solve(Ehat22) %*%
                  (x2 - muhat2)
                x <- X[j, ]
                x[mis] <- x1est
                T1 <- T1 + x  #est complete data contribution to the sufficient stat T1
                x1x1Test <- Ehat11 - Ehat12 %*% solve(Ehat22) %*%
                  Ehat21 + x1est %*% t(x1est)
                xxT <- X[j, ] %*% t(X[j, ])
                xxT[mis, mis] <- x1x1Test
                xxT[!mis, mis] <- x2 %*% t(x1est)
                xxT[mis, !mis] <- x1est %*% t(x2)
```

```
                T2 <- T2 + xxT  #est complete data contribution to the sufficient stat T2
            }
        }
        # save initial estimate to compute mre
        muhat_old <- muhat
        Ehat_old <- Ehat

        # Step two: Estimate
        muhat <- 1/n * T1
        Ehat <- 1/n * T2 - muhat %*% t(muhat)

        # calculate mre
        par_old <- c(muhat_old, Ehat_old[lower.tri(Ehat_old,
            diag = TRUE)])
        par_new <- c(muhat, Ehat[lower.tri(Ehat, diag = TRUE)])
        mre <- sqrt(sum((par_old - par_new)^2))/max(1, sqrt(sum(par_new^2)))
    }
    return(list(muhat, Ehat))
}


mu_start <- apply(X, 2, mean, na.rm = TRUE)
Xtemp <- X
Xtemp[1, 1] <- mu_start[1]
Xtemp[3, 3] <- mu_start[3]
E_start <- cov(Xtemp)

EM_normal(X, mu0 = mu_start, E0 = E_start)
```

```
## [[1]]
## [1]  4.4594571 -0.5545532  0.7703368
##
## [[2]]
##           [,1]      [,2]      [,3]
## [1,] 14.930346 11.245574  5.851375
## [2,] 11.245574 10.601760  9.078084
## [3,]  5.851375  9.078084 12.528188
```

Use a modified EM algorithm in which we update Σ from the entire dataset.

```
EM_normal_alt<-function(data,mu0,E0,tol=1e-5){

  muhat<-mu0
  Ehat<-E0
  X<-data
  n<-nrow(X)
  p<-ncol(X)
  mre<-10000 #modified relative error - something large/arbitrary to enter the while loop
  count<-0
    while(mre>tol){
      count<-count+1
      #Step one: Predict
      T1<-0
      X_temp<-X #data with missing data imputed with muhat of current iteration
                #- used to estimate E all at once
```

```r
      for(j in 1:n){
        if(sum(is.na(X[j,]))==0){
          x<-X[j,]
          T1<-T1+x
          next
        }else{
          mis<-is.na(X[j,])
          x1<-X[j,mis] #missing components
          x2<-X[j,!mis] #not missing
          #we estimate x1 with the conditional expectation of x1 given x2
          muhat1<-muhat[mis]
          muhat2<-muhat[!mis]
          Ehat11<-Ehat[mis,mis]
          Ehat12<-Ehat[mis,!mis]
          Ehat21<-Ehat[!mis,mis]
          Ehat22<-Ehat[!mis,!mis]
          x1est<-muhat1 + Ehat12%*%solve(Ehat22)%*%(x2-muhat2)
          x<-X[j,]
          x[mis]<-x1est
          T1<-T1+x #estimated complete data contribution to the sufficient statistic T1

          X_temp[j,]<-x
          }
      }
      #save initial estimate to compute mre
      muhat_old<-muhat
      Ehat_old<-Ehat

      #Step two: Estimate
      muhat<- 1/n*T1
      Ehat<-cov(X_temp)

      #calculate mre
      par_old<-c(muhat_old,Ehat_old[lower.tri(Ehat_old,diag=TRUE)])
      par_new<-c(muhat,Ehat[lower.tri(Ehat,diag=TRUE)])
      mre<-sqrt(sum((par_old-par_new)^2))/max(1,sqrt(sum(par_new^2)))
    }
  return(list(muhat,Ehat))
}

EM_normal_alt(X,mu0=mu_start,E0=E_start)
```

```
## [[1]]
## [1]  4.4594644 -0.5545532  0.7703378
##
## [[2]]
##           [,1]     [,2]      [,3]
## [1,] 18.663024 14.05701  7.314264
## [2,] 14.057015 13.25220 11.347603
## [3,]  7.314264 11.34760 15.660230
```

Both algorithms are off the mark in terms of both parameters but they are close considering the small sample size. Both agree on their mle of μ but in terms of the mle of Σ, the second method produces an mle estimate that is element-wise larger than the first. With the exception of the variance of X_2, the elements of sigma are inflated for both estimates, the second more so. This bias may be a product of the fact that the Σ is not estimated separately and is instead tied to the estimation of μ. Based on this simple simulation, I would think that the first method is superior to the second.