Some of the functions in the ContaminatedMixt Package and the need of self-coded functions for m-step and e-step

Jorge Sanchez

2023-11-08

Some interesting functions in the ContamiantedMixt package to work with contaminated mixtures of normal distributions

There are some functions relevant to work with contaminated mixture of normal distributions such as **m.step**, **CNmixt**, and **CNpredict**. The m.step function receive as input the observations that conforms the training set, the model to be fitted, the group information, and lastly the number maximum of iterations to convert. The output of this function returns the estimates for $\{\mu_g, \Sigma_g, \Sigma_g^{-1}, \pi_g\}_{g=1}^G$ the mean,variance matrix, inverse variance-covariance matrix, and the posterior distribution for each group. However, this function does not return estimates for parameters $\{\alpha_g, \eta_g\}_{g=1}^G$ the percentage of non-contaminated observations and inflation factor for each group. The absence of estimates for these parameters that would be required for an e-step makes sensible an implementation of an m-step that returns these estimates.

Using m.step function to obtain parameter estimates

A contaminated normal distribution is simulated with parameters $\mu_1 = (0,0,0,0,0), \mu_2 = c(0,6,0,6,0), \Sigma_1 = \Sigma_2 = Diag(1,1,1,1,1), \alpha_1 = \alpha_2 = 0.8, \eta_1 = \eta_2 = 20$. The estimates obtained using **m.step** function for $\mu_1, mu_2, \Sigma_1, \Sigma_2$ are shown below:

```r
# ContaminatedMixt::m step

estimates <- ContaminatedMixt::m.step(Xtrain,modelname = "VVV",z = unmap(ltrain), mmax = 10)
estimates$mu
```

```
##          group 1     group 2
## X.1  0.025984042 -0.04124758
## X.2  0.035366985  6.09698916
## X.3  0.142564546 -0.05096010
## X.4  0.007511553  6.06721447
## X.5 -0.050443365 -0.08032938
```

```r
estimates$Sigma
```

```
## , , group 1
##
##              X.1          X.2         X.3         X.4        X.5
## X.1  5.483980645 -0.007319953 -0.02155353  0.06565655  0.3923604
## X.2 -0.007319953  5.284518917  0.24709451  4.19523166 -0.0743575
## X.3 -0.021553533  0.247094505  4.81017843 -0.03436649 -0.1191220
## X.4  0.065656548  4.195231659 -0.03436649  4.90607673  0.3239015
## X.5  0.392360366 -0.074357504 -0.11912200  0.32390148  5.4032726
##
## , , group 2
##
##            X.1        X.2         X.3        X.4         X.5
## X.1  5.7944536 -0.3085605 -0.24307557 -0.5665974  0.11351621
## X.2 -0.3085605  5.0566552  0.39727750  4.2454198 -0.44416313
## X.3 -0.2430756  0.3972775  5.25940158  0.4401436  0.06270131
## X.4 -0.5665974  4.2454198  0.44014359  5.4590473 -0.16288768
## X.5  0.1135162 -0.4441631  0.06270131 -0.1628877  6.16304995
```

Using CNmixt function to obtain parameter estimates

It is possible to obtain parameter estimates using **CNmixt** function for the same contaminated data set. The main difference between **m.step** and **CNmixt** is that the later return estimates for the percengate of non-contaminated samples and inflation factor in each group. The results are as follows:

```
# predict using the function fitCMN from ContaminatedMixt packages
# supervised version
fitCMN <- CNmixt(Xtrain,G,contamination = TRUE,model = "VVV",
                 initialization = "mixt",label = ltrain,
                 iter.max = 10)
```

```
##
## Estimating model VVV contaminated with G = 2:**********
##
## The model estimated is contaminated, with G = 2 group(s), and parsimonious structure VVV
```

```
fitCMN$models[[1]]$mu
```

```
##         group 1     group 2
## X.1  0.01036543 -0.00248160
## X.2 -0.01824546  6.02271121
## X.3  0.02025561 -0.01637649
## X.4 -0.01853796  6.00891130
## X.5  0.01317919 -0.05253549
```

```
fitCMN$models[[1]]$Sigma
```

```
## , , group 1
##
##              X.1          X.2         X.3        X.4         X.5
## X.1  1.067747977  0.008472186 -0.01330243 0.01137948  0.06494513
## X.2  0.008472186  1.001786623  0.05342568 0.79310237 -0.02213401
## X.3 -0.013302428  0.053425680  1.01910417 0.02634575  0.02199739
## X.4  0.011379479  0.793102371  0.02634575 0.97154961  0.02663291
## X.5  0.064945126 -0.022134007  0.02199739 0.02663291  1.03937093
##
## , , group 2
##
```

```
##              X.1         X.2          X.3         X.4         X.5
## X.1  1.063585213 -0.03906702 -0.007661918 -0.03689629  0.03066820
## X.2 -0.039067022  0.97548010 -0.010535342  0.82313780 -0.03967217
## X.3 -0.007661918 -0.01053534  1.018012272 -0.03290716 -0.03991438
## X.4 -0.036896286  0.82313780 -0.032907163  1.06031010  0.01198749
## X.5  0.030668198 -0.03967217 -0.039914383  0.01198749  1.01241222
```

```
fitCMN$models[[1]]$alpha
```

```
## [1] 0.7984016 0.7807442
```

```
fitCMN$models[[1]]$eta
```

```
## [1] 20.74514 20.98491
```

Using self-coded function to obtain parameter estimates

```
fitCMN_SelfCoded <- ModelAccuracy2(Xtrain,Xtest,ltrain,ltest,"VVV",niterations = 10)
```

```
##
##  iter= 1 ; diflog= NA
##  iter= 2 ; diflog= 730.516
##  iter= 3 ; diflog= 1693.073
##  iter= 4 ; diflog= 1730.656
##  iter= 5 ; diflog= 529.7421
##  iter= 6 ; diflog= 38.72534
##  iter= 7 ; diflog= 0.14501
##  iter= 8 ; diflog= 0.6851312
##  iter= 9 ; diflog= 0.2156189
##  iter= 10 ; diflog= 0.05165247
```

```
fitCMN_SelfCoded$mu[[10]]
```

```
##             [,1]         [,2]
## [1,]  0.01030930 -0.002896591
## [2,] -0.01862156  6.022093411
## [3,]  0.01956207 -0.016948707
## [4,] -0.01899146  6.008724301
## [5,]  0.01379937 -0.052116499
```

```
fitCMN_SelfCoded$sigma[[10]]
```

```
## , , 1
##
##              [,1]         [,2]        [,3]       [,4]        [,5]
## [1,]  1.061827531  0.008659826 -0.01327292 0.01096511  0.06449082
## [2,]  0.008659826  0.995376527  0.05383263 0.78815187 -0.02111048
## [3,] -0.013272918  0.053832635  1.01357602 0.02703352  0.02288585
## [4,]  0.010965106  0.788151868  0.02703352 0.96583127  0.02668961
## [5,]  0.064490819 -0.021110479  0.02288585 0.02668961  1.03230762
##
## , , 2
##
##              [,1]        [,2]        [,3]        [,4]        [,5]
## [1,]  1.04929715 -0.03808885 -0.00836061 -0.03528128  0.03132694
## [2,] -0.03808885  0.96329771 -0.01059624  0.81394387 -0.03768232
## [3,] -0.00836061 -0.01059624  1.00023416 -0.03365746 -0.03794000
## [4,] -0.03528128  0.81394387 -0.03365746  1.04832396  0.01236988
## [5,]  0.03132694 -0.03768232 -0.03794000  0.01236988  0.99684213
```

```
fitCMN_SelfCoded$alpha[[10]]
```

```
## [1] 0.7974377 0.7782293
```

```
fitCMN_SelfCoded$eta[[10]]
```

```
## [1] 20.78686 21.30661
```

Using CNpredict function to obtain predictions

This function has a limitation that when the parameters α and η are passed, it returns an error. However, it works well when these parameters are set NULL values.

```
predCMN <- CNpredict(newdata = Xtest,
                     prior = fitCMN$models[[1]]$prior,
                     mu = fitCMN$models[[1]]$mu,
                     invSigma = fitCMN$models[[1]]$invSigma,
                     alpha = NULL,
                     eta = NULL)
predCMN
```

```
##   [1] 1 1 2 2 1 2 1 2 1 2 2 2 2 2 1 1 2 2 1 1 1 1 2 2 1 2 1 2 1 1 2 1 1 1 2 2 1
##  [38] 1 2 1 1 1 2 2 1 2 1 1 2 1 2 1 2 1 2 2 2 1 2 1 2 2 1 1 2 2 1 2 1 2 1 1 2 2
##  [75] 2 1 1 2 2 1 2 2 1 1 2 1 1 2 2 2 1 2 1 2 1 1 1 2 2 1 1 2 2 1 2 2 2 1 1 2 1
## [112] 1 2 2 1 1 1 2 1 2 2 2 1 2 2 2 1 1 1 2 1 2 1 2 2 1 2 1 2 1 2 1 1 1 2 1 2 1
## [149] 2 1 1 2 2 2 2 1 1 2 1 2 2 1 1 1 1 2 2 2 1 1 2 1 2 2 2 1 2 1 1 2 1 1 1 2 1
## [186] 2 2 2 1 2 2 1 1 2 1 2 1 2 1 2 1 2 2 2 2 2 1 1 1 1 1 2 2 1 2 2 1 2 1 2 1 2
## [223] 1 2 1 1 1 2 2 1 2 2 2 1 1 1 2 2 1 2 2 1 1 1 2 2 2 1 2 1 2 1 1 2 1 2 2 1 2
## [260] 1 2 1 2 1 2 1 2 1 2 1 1 2 2 2 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1
## [297] 2 2 1 1 1 2 1 1 2 2 1 2 1 1 1 2 1 2 2 2 2 2 2 1 2 1 1 1 1 1 1 1 1 2 1 1 1
## [334] 2 2 2 1 2 2 1 2 2 1 1 1 2 1 2 1 1 2 2 1 2 1 2 1 1 2 2 2 2 2 1 2 1 2 1 1 2
## [371] 2 1 2 2 1 2 2 2 1 1 2 1 2 2 1 2 1 2 2 1 2 2 1 1 1 1 1 2 1 2 1 1 1 2 2 2 1
## [408] 2 2 1 1 1 2 2 2 1 2 2 1 2 1 2 1 2 1 2 1 2 2 2 1 1 2 1 1 1 1 1 1 1 2 2 1 2
## [445] 1 2 1 1 2 2 1 1 1 2 2 1 2 2 1 1 2 1 2 2 1 2 1 2 2 2 2 1 1 1 1 2 1 1 1 2 2
## [482] 1 2 2 2 2 1 2 2 2 1 1 1 2 1 2 2 1 2 2 1 1 2 1 1 1 2 2 1 2 2 2 2 1 2 2 2 2
## [519] 2 1 2 2 1 2 1 1 2 2 1 2 1 2 2 1 1 2 1 1 2 2 1 1 1 2 1 1 1 2 1 1 1 2 2 2 2
## [556] 2 1 1 2 1 2 1 2 2 1 1 2 1 2 2 1 2 1 1 1 1 1 1 1 2 2 1 2 1 1 2 1 2 2 1 1 1
## [593] 1 1 1 1 1 1 1 1 2 2 2 1 2 1 1 1 1 2 2 1 2 2 1 2 2 1 2 1 2 1 1 2 2 2 1 2 2
## [630] 2 1 1 1 2 2 2 2 2 2 1 1 2 1 1 2 1 1 1 2 2 2 1 2 2 1 2 1 1 2 2 1 1 1 1 2 1
## [667] 1 2 1 2 2 1 2 2 2 2 2 1 2 1 1 2 2 1 1 1 1 1 2 2 1 1 2 1 2 2 1 1 1 1 1 2 1
## [704] 1 1 1 2 1 2 2 1 1 2 1 1 2 1 2 1 2 2 2 1 2 2 2 1 1 2 1 2 1 1 1 2 2 2 2 2 2
## [741] 2 2 1 1 1 1 1 1 2 2
```

Hence a E-step function is needed in order to predict contaminated observations.