EM algorithm for Multivariate Gaussian Mixture Model

Data import

```
iris = read.csv("D:/Projects/EM-algorithm/data/Iris.csv", header = TRUE, sep = ",")

iris2 = iris[,1:4]

print(iris2)
```

Algorithm implementation

```
#------------------------ Expectation-Maximization Algorithm -----------------------
gaussmixEM = function(params, X, clusters = 2, tol=.00001, maxits=100, showits=T){
  # Arguments:
  # params: list of values for mu, var, and probs
  # X: data matrix
  # clusters: number of clusters
  # tol: tolerance for convergence
  # maxits: maximum number of iterations
  # showits: print iterations or not

  require(mvtnorm)
  # Starting points
  N = nrow(X)
  mu = params$mu
  var = params$var
  probs = params$probs

  # initializations
  responsibility = matrix(0, ncol=clusters, nrow=N)
  logLikelihood = 0
  iteration = 0
  converged = FALSE

  if (showits)
    cat(paste("Iterations of EM:", "\n"))

  while (!converged & iteration < maxits) {
    probsOld = probs
    logLikelihoodOld = logLikelihood
    responsibilityOld = responsibility

#-------------------------------- Expectation-Step ---------------------------------
    # Compute responsibilities
    for (k in 1:clusters){
      responsibility[,k] = probs[k] * dmvnorm(X, mu[k,], sigma = var[[k]], log=F)
    }
    responsibility = responsibility/rowSums(responsibility)

#-------------------------------- Maximization-Step --------------------------------
    rk = colSums(responsibility)
    probs = rk/N
    for (k in 1:clusters){
      varMatrix = matrix(0, ncol=ncol(X), nrow=ncol(X))
      for (i in 1:N){
        varMatrix = varMatrix + responsibility[i,k] * X[i,]%*%t(X[i,])
      }
      mu[k,] = (t(X) %*% responsibility[,k]) / rk[k]
      var[[k]] =  varMatrix/rk[k] - mu[k,]%*%t(mu[k,])
      logLikelihood[k] = -.5 * sum( responsibility[,k] * dmvnorm(X, mu[k,],
sigma = var[[k]], log=T) )
    }
    logLikelihood = sum(logLikelihood)

    ### compare old to current for convergence
    paramsOld =  c(logLikelihoodOld, probsOld)
    paramsCurrent = c(logLikelihood, probs)
    iteration = iteration + 1
    if (showits & iteration == 1 | iteration%%5 == 0)
      cat(paste(format(iteration), "...", "\n", sep = ""))
    converged = min(abs(paramsOld - paramsCurrent)) <= tol
  }

  cluster = which(round(responsibility)==1, arr.ind=T)
  cluster = cluster[order(cluster[,1]), 2]
  out = list(probs=probs, mu=mu, var=var, resp=responsibility, cluster=cluster,
logLikelihood=logLikelihood)
}
```

Run EM algorithm

```
library(plyr)

# Create starting values
initMu = daply(iris, 'Species', function(x) colMeans(x[,1:4])) + runif(4, 0, .5)
initCov = dlply(iris, 'Species', function(x) var(x[,1:4]) + diag(runif(4, 0, .5)))
initProbs = c(.1, .2, .7)

initParams = list(mu=initMu, var=initCov, probs=initProbs)

# Run and examine
test = gaussmixEM(params=initParams, X=as.matrix(iris2), clusters = 3, tol=1e-8,
maxits=1500, showits=T)
```

```
## Loading required package: mvtnorm
```

```
## Iterations of EM:
## 1...
## 5...
```

```
table(test$cluster, iris$Species)
```

```
##    
##     Iris-setosa Iris-versicolor Iris-virginica
##   1          50               0              0
##   2           0              49              4
##   3           0               1             46
```

Results visualization

```r
library(ggplot2)

png('D:/Projects/EM-algorithm/plots/em-mgmm-plot-iris-example.png', width=1080,
height=2160)

# Scatter plot of clusters
ggplot(data = iris, aes(x = iris2$PetalLengthCm, y = iris2$PetalWidthCm,
color = as.factor(test$cluster))) +
  geom_point() +
  geom_point(data = as.data.frame(test$mu), aes(x = test$mu[,3], y = test$mu[,4]),
  color = "black", size = 4) +
  labs(title = "Scatter Plot of Clusters and Centroids", x = "Sepal Length",
  y = "Sepal Width") +
  theme_minimal()

dev.off()
```