EM algorithm for Gaussian Mixture Model

Data import and visualization

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

Algorithm implementation

```
#------------------------------- Expectation-Step ---------------------------------
expectation <- function(data_sample, p, mu_1, mu_2, var_1, var_2) {
  p_expectation <- (p * dnorm(data_sample, mean = mu_2, sd = sqrt(var_2))) /
 (p * dnorm(data_sample, mean = mu_2, sd = sqrt(var_2)) + (1 - p) * dnorm(data_sample,
mean = mu_1, sd = sqrt(var_1)))
  return(p_expectation)
}

#------------------------------- Maximization-Step --------------------------------
maximization <- function(sample,epart) {
  mu_1 <- sum((1 - epart) * sample) / sum(1 - epart)
  mu_2 <- sum(epart * sample) / sum(epart)
  var_1 <- sum((1 - epart) * (sample - mu_1)^2) / sum(1 - epart)
  var_2 <- sum(epart * (sample - mu_2)^2) / sum(epart)
  p <- sum(epart) / length(sample)
  return(c(p, mu_1, mu_2, var_1, var_2))
}

#------------------------ Expectation-Maximization Algorithm ----------------------
EM <- function(Y, parameters, iter, tol=1e-6){
  # Flag to check convergence
  flag <- 0
  # Responsiblity
  responsibility = matrix(0, nrow=length(Y))
  for(j in 1:iter) {
    for(i in 1:length(Y)) {
      responsibility[i] = expectation(Y[i], parameters[1], parameters[2], parameters[3],
parameters[4], parameters[5])
    }
    old_parameters = parameters
    parameters = maximization(Y, responsibility)

    if(all(abs(parameters - old_parameters) < tol)) {
      flag <- 1
      break
    }
  }
  if(!flag) {
    warning("Stopped at iteration ", j, "\nDidn't converge\n")
  }

  return(parameters)
}
```

Sample choice

```
Y = myData$SepalLengthCm
```

Data visualization

```
plot(hist(Y, breaks = 32))
```

Run EM algorithm

```
N=length(Y)
y=as.matrix(Y, nrow=N)

# Get Basic summary statistics
# Initalize first guess
# Mixing parameter pi
# Use random sample for random mu, estimator for variance

ybar=mean(Y)
pi=0.5
mu1=sample(Y, 1)
mu2=sample(Y, 1)
var1=sum(((Y - ybar)^2) / N)
var2=sum(((Y - ybar)^2) / N)

# initial guess (random)
init_guess=c(pi, mu1, mu2, var1, var2)

# Run on iterations
runEM = EM(Y, init_guess, iter=1000)
```

Results visualization

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

hist(Y, prob=T, breaks=32, Ylim=c(range(Y)[1], range(Y)[2]), main='')
```

```
## Warning in plot.window(xlim, ylim, "", ...): "Ylim" is not a graphical
## parameter
```

```
## Warning in title(main = main, sub = sub, xlab = xlab, ylab = ylab, ...): "Ylim"
## is not a graphical parameter
```

```
## Warning in axis(1, ...): "Ylim" is not a graphical parameter
```

```
## Warning in axis(2, at = yt, ...): "Ylim" is not a graphical parameter
```

```
lines(density(Y), col="#346e90", lwd=2)
x1 <- seq(from=range(Y)[1], to=range(Y)[2], length.out=length(Y))
y <- runEM[1] * dnorm(x1, mean=runEM[2], sd=sqrt(runEM[4])) + runEM[1] * dnorm(x1,
mean=runEM[3], sd=sqrt(runEM[5]))

lines(x1, y, col="#7d3333", lwd=2)

legend('topright', col=c("#346e90", '#7d3333'), lwd=2, legend=c("density", "fitted"))
```

```
dev.off()
```