Lab 6 - Computational Statistics (732A89)

Helena Llorens Lluís (hllor282), Yi Yang (yiyan338)

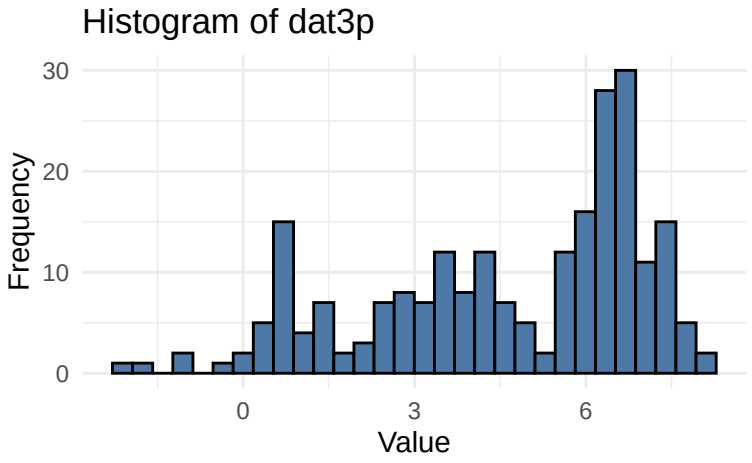
QUESTION 1: EM algorithm

Expectation-Maximization (EM) is an iterative optimization algorithm used for parameter estimation in probabilistic models with latent variables. In this report, we modify the provided EM algorithm to handle a mixture of three normal distributions.

Starting from the provided two-component EM algorithm, we introduced the following changes:

- Increased the number of mixture components from two to three.
- Adjusted the initialization of parameters accordingly.
- Modified the Expectation (E) step to compute probabilities for three components.
- Updated the Maximization (M) step to re-estimate parameters based on three components.
- Revised the stopping criterion to ensure scale-invariance by normalizing the convergence measure.

After loading the `dat3p` dataset, we plot a histogram to visualize its distribution.



The data exhibits multimodal behavior, suggesting the presence of multiple underlying distributions.

Then, we applied the EM algorithm with a stopping threshold of eps = 0.000001. The final estimated parameters of the normal mixture model are showed on the following table.

Table 1: Final estimated parameters of the normal mixture model

	p	mu	sigma
Component 1	0.3287399	2.337124	1.9199766
Component 2	0.2143430	3.979460	1.7707219
Component 3	0.4569171	6.627087	0.5635485

These values suggest that the algorithm successfully identified three distinct normal distributions underlying the dataset.

To assess convergence, we plotted the estimated μ and σ values across iterations.

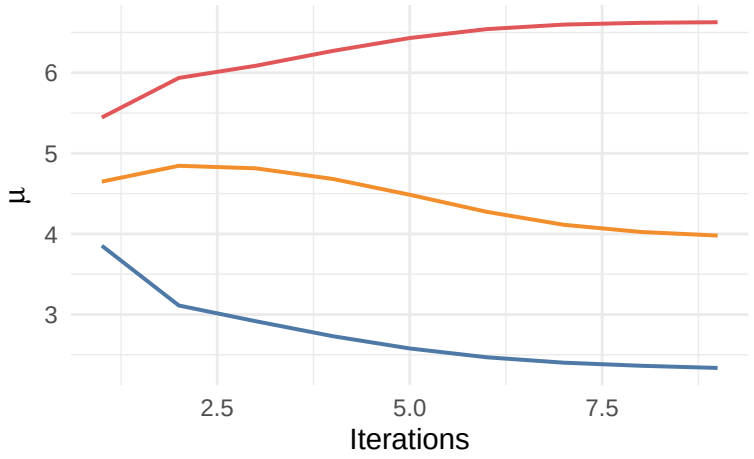


Figure 1: Plot of mu vs iteration number

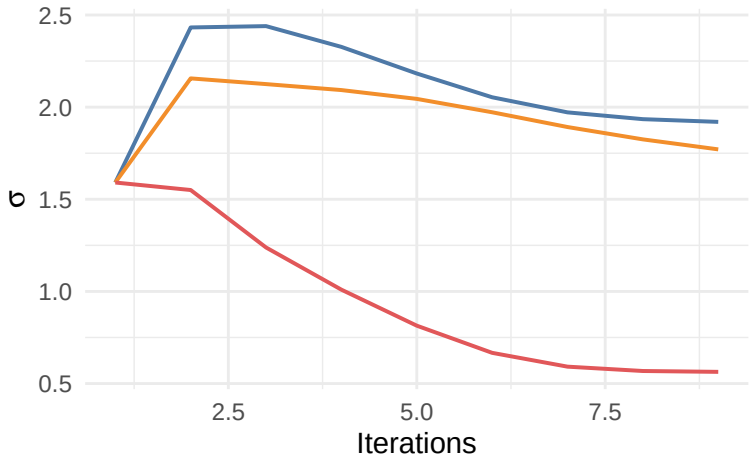


Figure 2: Plot of sigma vs iteration number

Both estimates stabilize after a certain number of iterations, supporting convergence.

The EM algorithm successfully fitted a three-component normal mixture model. The convergence plots validate that the algorithm converged for each parameter. Additionally, modifying the stopping criterion ensured robustness against data scaling.

Question 2 Simulated annealing

Fisrt, we plot the data and select randomly 22 clients.The red points show the initial cilents that we have choosen, which will be used in the simulated algorithm.

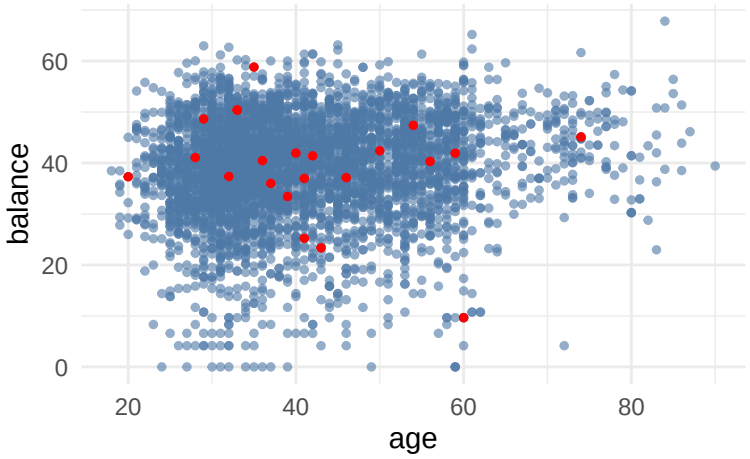


Figure 3: Plot of data

We now implement `simulated annealing` algorithm to minize the `criterion function`.To ensure we always have subsets with 22 clients, we **remove one client from the subsets and randomly select another client from the whole dataset in each iteration**.

We then try different combinations of schedules, temperatures and iterations to find a better solution.The fist combination is :

$$initialtemperature = 200$$

$$maxstage = 10$$

$$\alpha = 0.9$$

$$\beta = 1.5$$

$$initialiteration = 100$$

The plots of the criterion-value versus the iteration number ,as well as the final clients marked are shown:

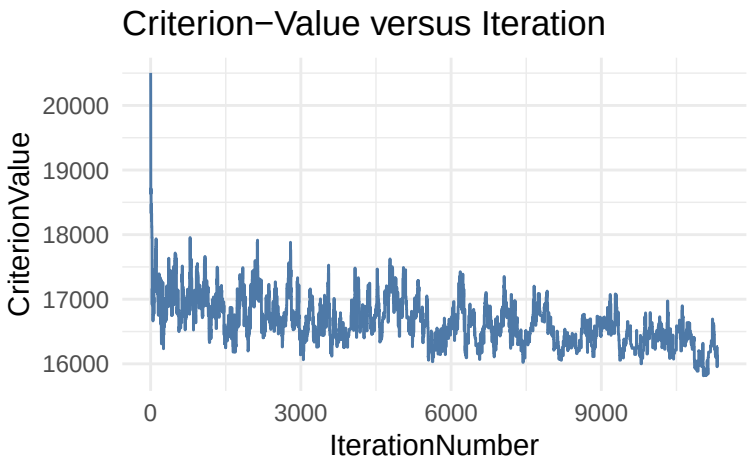


Figure 4: Criterion-value Versus the Iteration Number

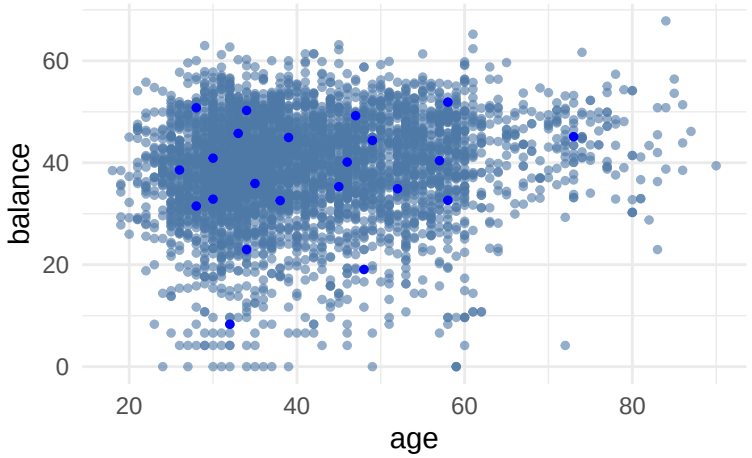


Figure 5: Final subset

The best crit value is:

```
## [1] 15815.4
```

Then we try the second combination:

$$initialtemperature = 200$$

$$maxstage = 10$$

$$\alpha = 0.9$$

$$\beta = 1.5$$

$$initialiteration = 10$$

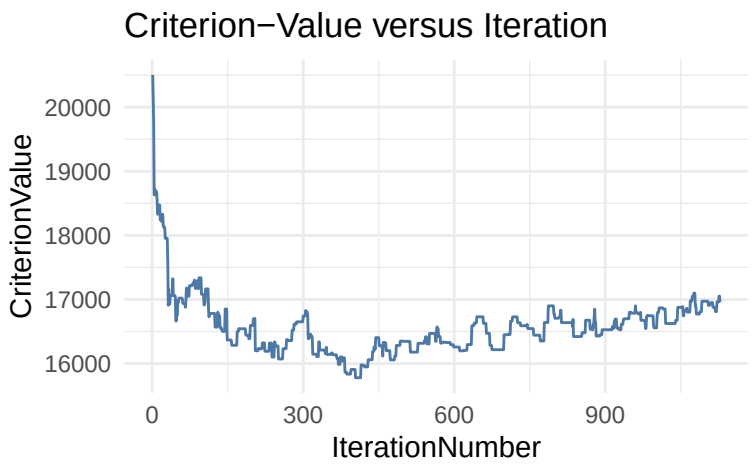


Figure 6: Criterion-value Versus the Iteration Number

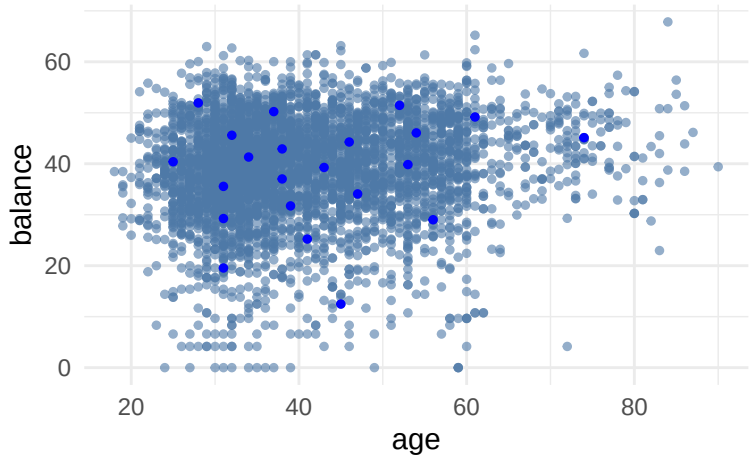


Figure 7: Final subset

The best crit value is:

```
## [1] 15778.63
```

Comparison
1.Solution quality:
The second combination achieved a lower criterion value (15778.63) compared to the first combination (15815.4).
2.Convergence Speed:
The first combination required a significantly higher number of iterations (9000) to reach the global minimum, indicating a slower convergence.
The second combination reached its best value much faster, around 370 iterations, indicating a quicker convergence.
3.Parameter Impact:
The higher initial iteration count (100) in the first combination allowed for more extensive exploration in each stage, which likely contributed to finding a better solution but at the cost of more iterations.

The lower initial iteration count (10) in the second combination led to faster convergence, allowing the algorithm to quickly find a better solution.

Conclusion
The second combination outperforms the first combination in both solution quality and convergence speed. This suggests that a lower initial iteration count with a higher iteration increase factor can lead to quicker and better solutions in this context.

APPENDIX

Question 1

```
library(ggplot2)
library(knitr)

# EM algorithm
em <- function(data, eps = 0.0001){
  n      <- length(data)
  pi     <- matrix(NA, ncol = 3, nrow = n)

  # Random initialization of the parameters
  p <- rep(1/3, 3)
  sigma <- rep(sd(data)*2/3, 3)
  mu <- c(mean(data) - sigma[1]/2, mean(data), mean(data) + sigma[1]/2)
  pv <- c(p, mu, sigma)
  cc  <- eps + 100
  mu_list <- c(mu)
  sigma_list <- c(sigma)

  while (cc > eps){
    # save previous parameter vector
    pv1  <- pv

    # E step
    for (j in 1:n){
      pi1 <- p[1]*dnorm(data[j], mean=mu[1], sd=sigma[1])
      pi2 <- p[2]*dnorm(data[j], mean=mu[2], sd=sigma[2])
      pi3 <- p[3]*dnorm(data[j], mean=mu[3], sd=sigma[3])

      total <- pi1+pi2+pi3
      pi[j,] <- c(pi1/total, pi2/total, pi3/total)
    }

    # M step
    p <- colSums(pi)/n
    mu[1] <- sum(pi[, 1]*data)/(p[1]*n)
    mu[2] <- sum(pi[, 2]*data)/(p[2]*n)
    mu[3] <- sum(pi[, 3]*data)/(p[3]*n)
    sigma[1] <- sqrt(sum(pi[, 1]*((data-mu[1])^2))/(p[1]*n))
    sigma[2] <- sqrt(sum(pi[, 2]*((data-mu[2])^2))/(p[2]*n))
    sigma[3] <- sqrt(sum(pi[, 3]*((data-mu[3])^2))/(p[3]*n))
```

```
    # stopping criteria
    pv <- c(p, mu, sigma)
    cc     <- sum((pv - pv1)^2) / sum(pv1^2)  # a convergence criterion, maybe not the best one
    mu_list <- rbind(mu_list, mu)
    sigma_list <- rbind(sigma_list, sigma)

  }
  res <- list(final_probs = pv[1:3],
              final_mus = pv[4:6],
              final_sigmas = pv[7:9],
              mu = mu_list,
              sigma = sigma_list)
  return(res)
}

# load data
load(file = "D:\\liu\\CompStat\\Computational-Statistics\\lab6\\threepops.Rdata")
data <- dat3p

# Histogram of the data
df <- data.frame(data)
ggplot(df, aes(x = df[, 1])) +
  geom_histogram(col = "black", fill = "#4E79A7") +
  labs(title = "Histogram of dat3p", x = "Value", y = "Frequency") +
  theme_minimal()

# Run algorithm on the data
res <- em(data)
final_probs <- res$final_probs
final_mus <- res$final_mus
final_sigmas <- res$final_sigmas

# results table
result_table <- data.frame(final_probs, final_mus, final_sigmas)
colnames(result_table) <- c("p", "mu", "sigma")
rownames(result_table) <- c("Component 1", "Component 2", "Component 3")
kable(result_table,
      caption = "Final estimated parameters of the normal mixture model")

#  plots of current estimates for each model parameter versus the iteration-number
# mu plot
mus <- data.frame(res$mu)
ggplot(data = mus, aes(x = 1:nrow(mus))) +
  geom_line(aes(y = mus[, 1]), col = "#4E79A7", size = 0.7) +
  geom_line(aes(y = mus[, 2]), col = "#F28E2B", size = 0.7) +
  geom_line(aes(y = mus[, 3]), col = "#E15759", size = 0.7) +
  theme_minimal() +
  labs(x = "Iterations", y = expression(mu))

# sigma plot
sigmas <- data.frame(res$sigma)
ggplot(data = sigmas, aes(x = 1:nrow(sigmas))) +
  geom_line(aes(y = sigmas[, 1]), col = "#4E79A7", size = 0.7) +
```

```
  geom_line(aes(y = sigmas[, 2]), col = "#F28E2B", size = 0.7) +
  geom_line(aes(y = sigmas[, 3]), col = "#E15759", size = 0.7) +
  theme_minimal() +
  labs(x = "Iterations", y = expression(sigma))
```

Question 2

```
  #question2
library(ggplot2)
load("bankdata.Rdata")
data <- as.data.frame(bankdata)
nclients <- dim(data)[1]  # number of individuals in the dataset, here 4364


crit <- function(dat, subs){
  s <- length(subs)
  dist <- matrix(rep(NA, nclients*s), ncol=s)
  for (i in 1:s){
    dist[, i] <- sqrt((dat[,1]-dat[subs[i],1])^2+(dat[,2]-dat[subs[i],2])^2)
  }
  sum(apply(dist, 1, min))
}


#a
set.seed(12345)
initial <- sample(1:nclients,22)
df <- data.frame(age=data$age,balance=data$balance)
ggplot(df,aes(age,balance))+
  geom_point(color="#4E79A7",size=2,alpha=0.6)+
  geom_point(data=df[initial,],aes(age,balance),color="red",size=2)+
  theme_minimal()

#b simulated annealing



exchange_clients <- function(initial,nclients){

  remove_index <- sample(1:length(initial),1)
  #select one sample from dataset
  add_index <- sample(setdiff(1:nclients,initial),1)
  new_initial <- initial
  new_initial[remove_index] <- add_index
  return(new_initial)

}

simulated_annealing <- function(data,initial,max_stage,initial_temperature,alpha,initial_iteration,beta

  current_subsample <- initial
```

```
current_crit <- crit(data,current_subsample)
best_subsample <- current_subsample
best_crit <- current_crit

temperature <- initial_temperature
iter_per_stage <- initial_iteration
crit_values <- numeric()
total_iter <- 0

for(stage in 1:max_stage){

  for (i in 1:iter_per_stage) {

    total_iter <- total_iter + 1
    #generate new subsample
    new_subsample <- current_subsample
    remove_index <- sample(1: length(current_subsample),1)
    #select one sample from dataset
    add_index <- sample(setdiff(1:nclients,current_subsample),1)
    new_subsample[remove_index] <- add_index

    #calculate new crit
    new_crit <- crit(data,new_subsample)

    #calculate the acceptance probability
    differernce_crit <- new_crit- current_crit
    accept_probability <- ifelse(differernce_crit<0,1,exp(-differernce_crit/temperature))

    if(runif(1)<accept_probability){
      current_subsample <- new_subsample
      current_crit <- new_crit

    }

    if(current_crit<best_crit){
      best_subsample <- current_subsample
      best_crit <- current_crit
    }

    crit_values[total_iter] <- current_crit
  }

    temperature <- temperature*alpha
    iter_per_stage <- iter_per_stage*beta


}

return(list(
  best_subsample=best_subsample,
  best_crit=best_crit,
  crit_values=crit_values
))
```

```
}


set.seed(12345)
result1 <- simulated_annealing(data=data,initial=initial,max_stage=10,initial_temperature=200,alpha=0.9
#plot(result1$crit_values,type = "l",col="red",xlab="Iteration Number",ylab="Criterition Value",main="C

dfcritvalue1 <- data.frame(IterationNumber=1:length(result1$crit_values),
                           CriterionValue=result1$crit_values)
ggplot(dfcritvalue1,aes(x=IterationNumber,y=CriterionValue))+
  geom_line(color="#4E79A7")+
  labs(x="IterationNumber",
       y="CriterionValue",
       title = "Criterion-Value versus Iteration")+
  theme_minimal()

dfresult1 <- data[result1$best_subsample, ]
ggplot(df,aes(age,balance))+
  geom_point(color="#4E79A7",size=2,alpha=0.6)+
  #geom_point(data=df[initial,],aes(age,balance),color="red",size=2)+
  geom_point(data=dfresult1,aes(age,balance),color="blue",size=2)+
  theme_minimal()

print(result1$best_crit)

set.seed(12345)
result2 <- simulated_annealing(data=data,initial=initial,max_stage=10,initial_temperature=200,alpha=0.9

dfcritvalue2 <- data.frame(IterationNumber=1:length(result2$crit_values),
                           CriterionValue=result2$crit_values)
ggplot(dfcritvalue2,aes(x=IterationNumber,y=CriterionValue))+
  geom_line(color="#4E79A7")+
  labs(x="IterationNumber",
       y="CriterionValue",
       title = "Criterion-Value versus Iteration")+
  theme_minimal()

dfresult2 <- data[result2$best_subsample, ]
ggplot(df,aes(age,balance))+
  geom_point(color="#4E79A7",size=2,alpha=0.6)+
  #geom_point(data=df[initial,],aes(age,balance),color="red",size=2)+
  geom_point(data=dfresult2,aes(age,balance),color="blue",size=2)+
  theme_minimal()
print(result2$best_crit)
```