Relatório trabalho prático 4

César A. Galvão 19/0011572 Gabriela Carneiro 18/0120816

11 de September de 2022

Contents

1 Resumo

Nesta atividade foram implementadas em R um algoritmo EM e um método MCMC. Para cada algoritmo, foram estimados três parâmetros: probabilidade de o lote vir da máquina A (p) e a probabilidade de que, dado que uma determinada peça tenha sido produzida pela máquina A ou B, a peça seja defeituosa (θ_A, θ_B). Por fim, são apresentados histogramas com a função de probabilidade conjunta obtida pelos parâmetros estimados sobreposta em formato de pontos.[1]

[1] Todos os documentos desse relatório podem ser verificados no repositório https://github.com/cesar-galvao/Estatistica-computacional

2 Método

2.1 Método EM

Na presença de variáveis ausentes ou outras condições semelhantes, EM (Expectation-Maximization) é um algoritmo seguro que garante a convergência para ao menos um máximo local. Cada iteração do algoritmo EM consiste em duas etapas. Primeiramente, a etapa *Expectation* (E), consiste em atribuir uma distribuição de probabilidades para os complementos dos dados, dado o modelo atual. Em seguida, os parâmetros do modelo são reestimados (Etapa M). Este procedimento iterativo é repetido até que algum critério de convergência seja satisfeito.

O algoritmo EM tem duas vantagens principais que o torna uma boa escolha para solucionar problemas de máxima verossimilhança:

1. O algoritmo EM fornece condição para satisfazer automatecamente restrições probabilísticas de *mixture models*, que são modelos probabilísticos para representar a presença de subpopulações dentro de uma população maior, sem que seja necessário que haja um conjunto de dados observado para identificar a subpopulação à qual uma observação individual pertence. Outras abordagens de otimização exigem mais custos computacionais e o desenvolvimentos de algoritmos mais complexos.
2. O algoritmo EM garante um sequência crescente de verossimilhança e uma convergência monotônica segura.

É importante ressaltar que a seleção de valores iniciais é um passo muito importante para este procedimento iterativo, uma vez que a seleção apropriada de valores iniciais pode aumentar a velocidade de convergência. Mais importante ainda, a escolha dos valores iniciais auxilia o algoritmo a evitar ótimos locais para alcançar uma solução global.

2.2 Amostrador de Gibbs

O amostrador de Gibbs é um dos tipos de método de simulação de Monte Carlo via cadeia de Markov.

A ideia por trás desses métodos é simples e geral. Para amostrar uma determinada distribuição de probabilidade, referida como a distribuição alvo, uma cadeia de Markov adequada é construída com a propriedade de que sua distribuição limitante e invariante seja a própria distribuição alvo. Dependendo das especificidades do problema, a cadeia de Markov pode ser construída pelo algoritmo de Metropolis-Hastings (M-H), sendo o método de amostragem de Gibbs um caso especial do método Metropolis, ou misturas desses dois algoritmos. Uma vez que a cadeia de Markov tenha sido construída, uma amostra da distribuição alvo pode ser obtida simulando a cadeia de Markov um grande número de vezes e registrando seus valores. Em muitas situações, método de simulação de Monte Carlo via cadeia de Markov fornece a única maneira prática de obter amostras de distribuições de probabilidade de dimensões superiores.

No algoritmo de amostragem de Gibbs os parâmetros são agrupados em p blocos $(\psi_1, ..., \psi_p)$ e cada bloco é amostrado de acordo com a distribuição condicional completa do bloco ψ_k, definida como a distribuição condicional sob π de ψ_k dado todos os outros blocos ψ_{-k} e denotada como $\pi(\psi_k|\psi_{-k})$. Em paralelo com o algoritmo M-H de vários blocos, o valor mais atual dos blocos restantes é usado para derivar a distribuição condicional completa de cada bloco.

3 Resultados

Para o método EM, obteve-se a seguinte estimativa final para os parâmetros ao final de 29 iterações, considerando o critério de parada $|\gamma_i^{(k+1)} - \gamma_i^{(k)}| < 10^{-9}$:

θ_A	θ_B	p
0.0517882	0.0905156	0.6908754

A função de distribuição conjunta, com os parâmetros obtidos pelo algoritmo EM, é definida

$$P(y) = (1-p)P(y|\theta_A) + pP(y|\theta_B). \tag{1}$$

Além disso, obteve-se o seguinte histograma com a função de distribuição conjunta, representada por pontos:

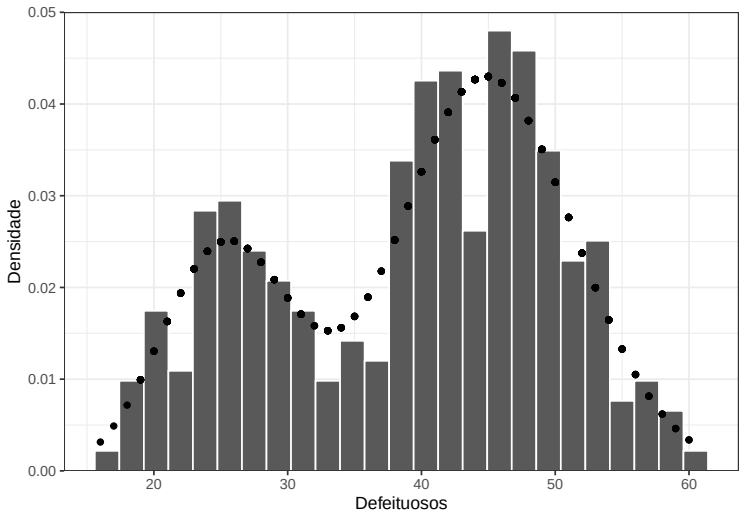

Figure 1: Histograma de da amostra e função de distribuição conjunta com parâmetros obtidos pelo algoritmo EM.

Para o amostrador de Gibbs, as seguintes foram as estimativas para os parâmetros:

θ_A	θ_B	p
0.0905063	0.0518119	0.3099991

Obteve-se o seguinte histograma com a mesma função de distribuição conjunta (1), porém utilizando as novas estimativas, representada por pontos:

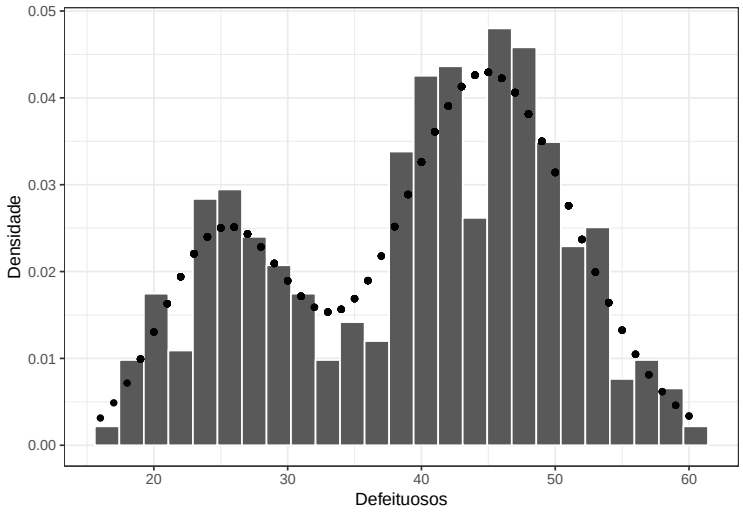

Figure 2: Histograma de da amostra e função de distribuição conjunta com parâmetros obtidos pelo Amostrador de Gibbs.

Finalmente, compara-se os métodos. Obtém-se uma diferença máxima de 0.00006 entre estimativas dos métodos, indicando pouca diferença no cenário considerado – duas subpopulações com medidas descritas por variáveis binomiais.

O gráfico a seguir sugere uma certa homogeneidade das diferenças, o que a princípio não sugeriria a preferência por um método específico para esse conjunto de dados.

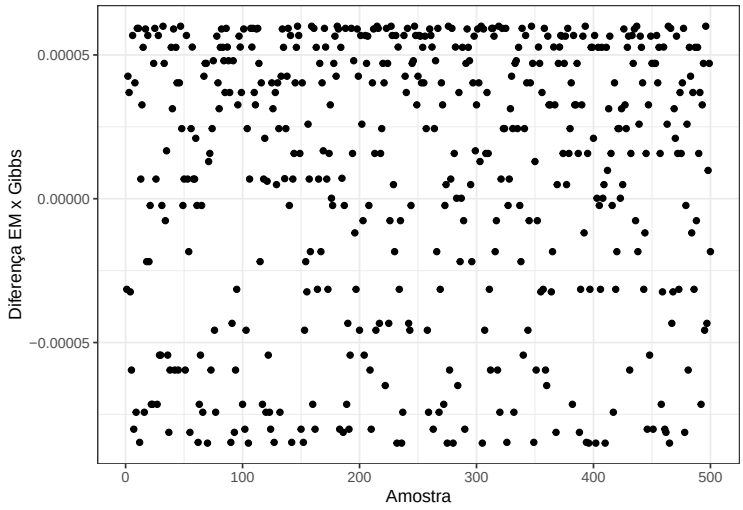

Figure 3: Dispersão da diferença de estimação entre os métodos EM e Amostrador de Gibbs.

Anexo A - código comentado

```
# entrada de dados ----
library(tidyverse)

dados <- read.csv2("data.csv",
                   col.names = c("lote", "defeituosos")) %>%
  mutate(prop = defeituosos/500)

# metodo EM ----

#inicializando thetaA e thetaB
set.seed(1234)
thetaA <- sample(dados$prop, 1)
thetaB <- sample(dados$prop, 1)
p <- runif(1)


## prepara dados de iteração com primeiro gamma ----
itera <- dados %>% #gamma k+1
  mutate(gamma1 = p*dbinom(defeituosos, size = 500, prob = thetaB)/
           ((1-p)*dbinom(defeituosos, size = 500, prob = thetaA)+
              p*dbinom(defeituosos, size = 500, prob = thetaB)),
         um_menos_gama1 = 1-gamma1)

itera$dif <- 1 #dif inicial para não travar o while

## inicia iterações ----

contador <- 0

while(any(itera$dif > 10^(-9)) == TRUE){

  contador <- contador+1

  itera$gamma2 <- itera$gamma1 #guarda o gama k

  ## M step ----

  thetaA <- sum(itera$defeituosos*(itera$um_menos_gama1))/(500*sum(itera$um_menos_gama1))
  thetaB <- sum(itera$defeituosos*(itera$gamma1))/(500*sum(itera$gamma1))
  p <- sum(itera$gamma1)/500

  ## E step ----

  itera <- itera %>%
    mutate(gamma1 = p*dbinom(defeituosos, size = 500, prob = thetaB)/
             ((1-p)*dbinom(defeituosos, size = 500, prob = thetaA)+
                p*dbinom(defeituosos, size = 500, prob = thetaB)),
           um_menos_gama1 = 1-gamma1)

  itera$dif <- itera$gamma1 - itera$gamma2
```

```
}

## funcao de probabilidade ----
Py <- function(y){
  return((1-p)*dbinom(y, size = 500, prob = thetaA)+ p*dbinom(y, size = 500, prob = thetaB))
}

## histograma ----

dados_plot <- data.frame(x = dados$defeituosos, y = Py(dados$defeituosos))

histograma_em <- ggplot(dados, aes(x = defeituosos))+
  geom_histogram(aes(y = ..density..),bins = 25, color = "white")+
  geom_point(data = dados_plot, aes(x = x, y = y))+
  labs(y = "Densidade", x = "Defeituosos")+
  scale_y_continuous(expand = c(0,0), limits = c(0, 0.05))+
  theme_bw()+
  theme(#panel.grid = element_blank()#,
        # panel.border = element_blank(),
        # axis.line = element_line()
        )


# Amostrador de Gibbs ----

## inicializando os parametros ----

thetaa <- rbeta(1,1,1)
thetab <- rbeta(1,1,1)
p2 <- rbeta(1,1,1)

## iterações ----

### prepara dados de iteração com primeiro delta ----
itera2 <- dados %>%
  mutate(delta1 = (p2*dbinom(defeituosos, size = 500, prob = thetab))/
           ((1-p2)*dbinom(defeituosos, size = 500, prob = thetaa)+
              p2*dbinom(defeituosos, size = 500, prob = thetab)),
         um_menos_delta1 = 1-delta1)


### inicia iterações ----

vec_thetaa <- c()
vec_thetab <- c()
vec_p <- c()

for(i in 1:20000){
  #### atualiza parametros ----

  alfaa <- 1+sum(itera2$defeituosos*itera2$um_menos_delta1)
  betaa <- 1+sum((500-itera2$defeituosos)*itera2$um_menos_delta1)
```

```r
  alfab <- 1+sum(itera2$defeituosos*itera2$delta1)
  betab <- 1+sum((500-itera2$defeituosos)*itera2$delta1)

  alfap <- 1+sum(itera2$delta1)
  betap <- 1+sum(itera2$um_menos_delta1)

  thetaa <- rbeta(1, alfaa, betaa)
  thetab <- rbeta(1, alfab, betab)
  p2 <- rbeta(1, alfap, betap)

  vec_thetaa[i] <- thetaa
  vec_thetab[i] <- thetab
  vec_p[i] <- p2


  #### atualiza delta ----

  itera2 <- itera2 %>%
    mutate(delta1 = p2*dbinom(defeituosos, size = 500, prob = thetab)/
             ((1-p2)*dbinom(defeituosos, size = 500, prob = thetaa)+
                p2*dbinom(defeituosos, size = 500, prob = thetab)),
           um_menos_delta1 = 1-delta1)
}

## media das S ultimas amostras ----

thetaa_final <- mean(vec_thetaa[10001:20000])
thetab_final <- mean(vec_thetab[10001:20000])
p2_final <- mean(vec_p[10001:20000])

## funcao de probabilidade ----
Py2 <- function(y){
  return((1-p2_final)*dbinom(y, size = 500, prob = thetaa_final)+
           p2_final*dbinom(y, size = 500, prob = thetab_final))
}

## histograma ----

dados_plot2 <- data.frame(x = dados$defeituosos, y = Py2(dados$defeituosos))

histograma_gibbs <- ggplot(dados, aes(x = defeituosos))+
  geom_histogram(aes(y = ..density..),bins = 25, color = "white")+
  geom_point(data = dados_plot2, aes(x = x, y = y))+
  labs(y = "Densidade", x = "Defeituosos")+
  #scale_y_continuous(expand = c(0,0), limits = c(0, 0.05))+
  theme_bw()+
  theme(#panel.grid = element_blank()#,
    # panel.border = element_blank(),
    # axis.line = element_line()
  )

# diferença entre os métodos ----
```

```r
diferencas <- data.frame(
  x = 1:500,
  amostra = dados$defeituosos,
  em = Py(dados$defeituosos),
  gibbs = Py2(dados$defeituosos)
) %>% mutate(diff = em-gibbs)

options(scipen = 99999)
maxdiff <- max(diferencas$diff) %>% round(5)

graf_diff <- ggplot(diferencas, aes(x = x, y = diff))+
  geom_point()+
  labs(y = "Diferença EM x Gibbs", x = "Amostra")+
  theme_bw()
```