Homework 3

(1) EM Algorithm

1) observed data vector: $X = (X_1, \dots, X_n)$

latent variable vector: $Z = (Z_1, \dots, Z_n)$, each Z_i 0 or 1

distribution of Z: $X_i | Z_i = 0 \sim Pois(\mu_0)$, $X_i | Z_i = 1 \sim Pois(\mu_1)$

$P(Z_i = 0) = \pi$, $P(Z_i = 1) = 1 - \pi$

2) E step: solve for $Q(\theta | \theta^{(t)})$ = estimating θ based on $\theta^{(t)}$ from tth iter, where $\theta = (\mu_0, \mu_1, \pi)$

$$Q(\theta|\theta^{(t)}) = E_{Z|X,\theta^{(t)}}\left(\log L(\theta|X,Z)\right) = E_{Z|X,\theta^{(t)}}\left(\sum_{i=1}^{n}\log L(\theta|X,Z)\right)$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{2} P(Z_i = j | X_i = x_i, \theta^{(t)}) \log L(\theta_j | x_i, j) \quad \rightarrow L(\theta_j | x_i, j) = \frac{\pi_j P_{\mu_j}(x_i)}{\pi_0 P_{\mu_0}(x_i) + \pi_1 P_{\mu_1}(x_i)}$$

$$= \sum_{i=1}^{n}\sum_{j=0}^{1} T_{j,i}^{(t)}\left(\log(\pi_j P_{\mu_j}(x_i)) - \log\left(\pi_0 P_{\mu_0}(x_i) + \pi_1 P_{\mu_1}(x_i)\right)\right)$$, where $\pi_0 = \pi$ & $\pi_1 = 1 - \pi$

here $T_{j,i}^{(t)} := P(Z_i = j | X_i = x_i, \mu_0^{(t)}, \mu_1^{(t)}, \pi^{(t)}) = \dfrac{\pi_j^{(t)} P_{\mu_j^{(t)}}(x_i)}{\pi_0^{(t)} P_{\mu_0^{(t)}}(x_i) + \pi_1^{(t)} P_{\mu_1^{(t)}}(x_i)}$

3) M step: $\pi^{(t+1)} = \underset{\pi}{\arg\max}\, Q(\theta|\theta^{(t)}) \overset{!}{=} \dfrac{1}{n}\sum_{i=1}^{n} T_{0,i}^{(t)}$

$$\mu_0^{(t+1)} = \underset{\mu_0}{\arg\max}\, Q(\theta|\theta^{(t)}) = \frac{1}{\sum_{i=1}^{n} T_{0,i}^{(t)}} \cdot \sum_{i=1}^{n} T_{0,i}^{(t)} x_i$$

$$\mu_1^{(t+1)} = \underset{\mu_1}{\arg\max}\, Q(\theta|\theta^{(t)}) = \frac{1}{\sum_{i=1}^{n} T_{1,i}^{(t)}} \cdot \sum_{i=1}^{n} T_{1,i}^{(t)} x_i$$

4) Use the result of k-means as an initial estimator

$$\pi^{(0)} = \frac{\#\text{ of clustered } 0}{n}, \quad \mu_0^{(0)} = \frac{\sum x_i \text{ clustered } 0}{\#\text{ clustered } 0}, \quad \mu_1^{(0)} = \frac{\sum x_i \text{ clustered } 1}{\#\text{ clustered } 1}$$

Homework 3

Sahil Saxena

10/27/2021

5

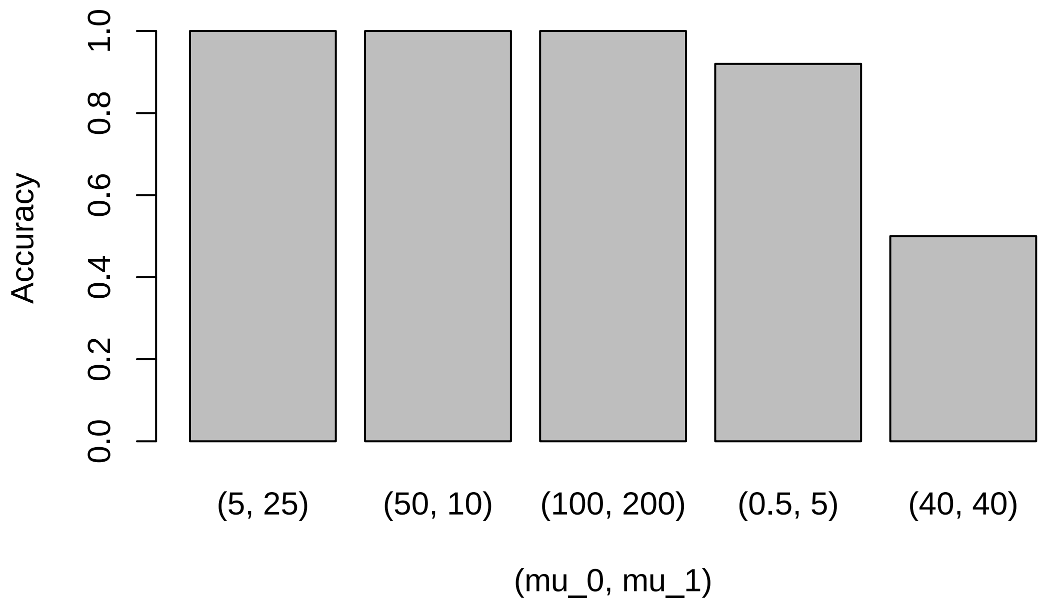
```
e_step <- function(x, z, pi, mu_0, mu_1) {
  # Input: x is the observation vector
  #        z is the hidden variable (0 or 1)
  #        pi, mu_0, mu_1 are the current parameters esitmates
  # Output: vector containing T_zi's

  # P0(x) and P1(x) with current parameter estimates
  p0_x <- dpois(x, lambda = mu_0)
  p1_x <- dpois(x, lambda = mu_1)

  # compute the term in the denominator
  denom <- pi * p0_x + (1 - pi) * p1_x

  # compute the term in the numerator
  if (z == 0)
    num <- pi * p0_x
  else
    num <- (1 - pi) * p1_x

  return(num / denom)
}

m_step <- function(x, T_0, T_1) {
  # Input: x is the observation vector
  #       T_0 is nx1 vector of T_0i's
  #       T_1 is nx1 vector of T_1i's
  # Output: vector containing new estimates of pi, mu_0, mu_1

  mu_0 <- sum(T_0 * x) / sum(T_0)
  mu_1 <- sum(T_1 * x) / sum(T_1)
  pi <- mean(T_0)
  return(c(pi, mu_0, mu_1))
}

initial_vals <- function(x, n) {
  # Input: x is the observation vector
  # Output: vector containing initial guesses for pi, mu_0, mu_1

  kmeans <- kmeans(x, centers = 2)
```

```r
  # always label the first distribution as "1"
  if (kmeans$centers[1] > kmeans$centers[2]) {
    new_label = kmeans$cluster
    new_label[kmeans$cluster == 1] = 2
    new_label[kmeans$cluster == 2] = 1
    kmeans$cluster <- new_label
  }

  pi <- sum(kmeans$cluster == 1) / n
  mu_0 <- mean(x[kmeans$cluster == 1])
  mu_1 <- mean(x[kmeans$cluster == 2])

  return(c(pi, mu_0, mu_1))
}

# return true if EM converged with threshold of 0.001
converged <- function(theta_prev, theta_curr) {
  dist <- abs(theta_prev - theta_curr) / theta_prev
  return(all(dist <  0.001))
}
```

6

```r
EM <- function(n, mu_0, mu_1) {
  # Input: n is the number of observations from each distribution
  #        so N in total is actually 2 * n
  #        mu_0, mu_1 are the true means used in simulations
  # Output: c(pi, mu_0, mu_1, accuracy), where accuracy is the
  #         assignment accuracy, and the first three elements are
  #         the estimates of the parameters from EM

  x_1 <- rpois(n, lambda = mu_0)
  x_2 <- rpois(n, lambda = mu_1)
  x <- c(x_1, x_2)

  init <- initial_vals(x, 2*n)

  theta_prev <- c(pi, mu_0, mu_1)
  T_0 <- e_step(x, 0, pi, mu_0, mu_1)
  T_1 <- e_step(x, 1, pi, mu_0, mu_1)
  theta_curr <- m_step(x, T_0, T_1)

  for (i in 1:5000) {
    if (converged(theta_prev, theta_curr)) break

    theta_prev <- theta_curr
    T_0 <- e_step(x, 0, theta_curr[1], theta_curr[2], theta_curr[3])
    T_1 <- e_step(x, 1, theta_curr[1], theta_curr[2],theta_curr[3])
    theta_curr <- m_step(x, T_0, T_1)
  }
  result <- theta_curr

  T_0 <- e_step(x, 0, pi, mu_0, mu_1)
```

```
  cluster <- rep(0, length(x))
  cluster[T_0 < .5] <- 1 # assign i to cluster 0 if T_0i > .5

  # measure of accuracy
  cluster_true <- c(rep(0, n), rep(1, n)) # true clusters
  accuracy <- sum(cluster == cluster_true) / (2*n)
  result <- c(result, accuracy)
  return(result)
}

p6_1 <- EM(50, 5, 25)
p6_2 <- EM(50, 50, 10)
p6_3 <- EM(50, 100, 200)
p6_4 <- EM(50, 0.5, 5)
p6_5 <- EM(50, 40, 40)
acc <- c(p6_1[4], p6_2[4], p6_3[4], p6_4[4], p6_5[4])
barplot(acc, names.arg=c("(5, 25)", "(50, 10)", "(100, 200)", "(0.5, 5)", "(40, 40)"), xlab="(mu_0, mu_
```

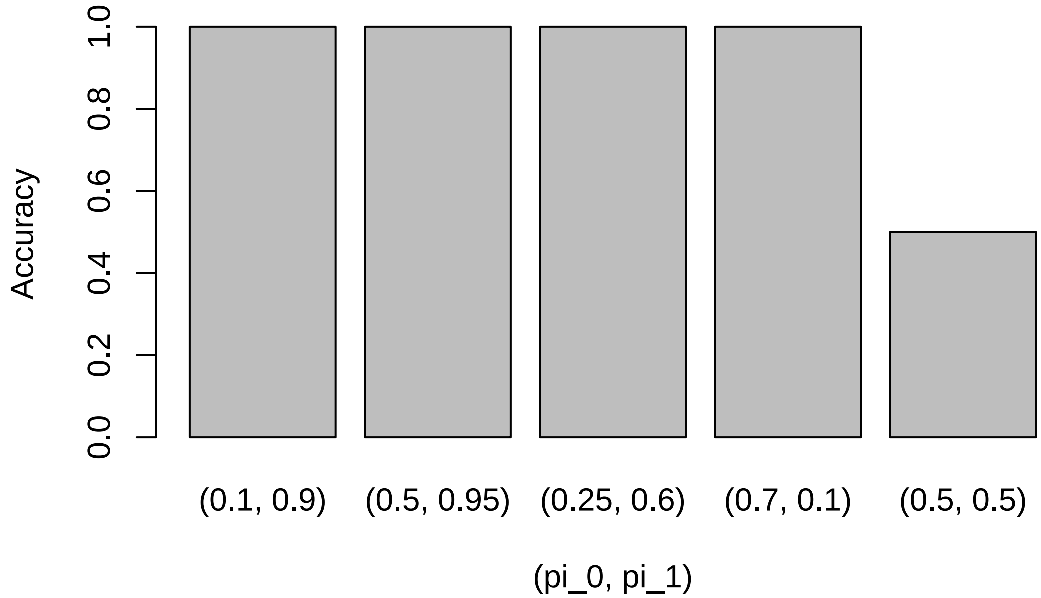


Bar chart shows that the accuracy of EM clustering with poisson distributions is very high. As expected, accuracy drops to 50% when the mixing distributions are identical (it's like choosing 50/50 every time).

7

```
EM2 <- function(n, pi_0, pi_1) {
  # Input: n is the number of observations from each distribution
  #        so N in total is actually 2 * n
  # Output: c(pi, mu_0, mu_1, accuracy), where accuracy is the
  #         assignment accuracy, and the first three elements are
  #         the estimates of the parameters from EM

  x_1 <- rbinom(n, size = 100, prob = pi_0)
  x_2 <- rbinom(n, size = 100, prob = pi_1)
  x <- c(x_1, x_2)
```

```
  mu_0 <- 100 * pi_0
  mu_1 <- 100 * pi_1
  init <- initial_vals(x, 2*n)

  theta_prev <- c(pi, mu_0, mu_1)
  T_0 <- e_step(x, 0, pi, mu_0, mu_1)
  T_1 <- e_step(x, 1, pi, mu_0, mu_1)
  theta_curr <- m_step(x, T_0, T_1)

  for (i in 1:5000) {
    if (converged(theta_prev, theta_curr)) break

    theta_prev <- theta_curr
    T_0 <- e_step(x, 0, theta_curr[1], theta_curr[2], theta_curr[3])
    T_1 <- e_step(x, 1, theta_curr[1], theta_curr[2],theta_curr[3])
    theta_curr <- m_step(x, T_0, T_1)
  }
  result <- theta_curr

  T_0 <- e_step(x, 0, pi, mu_0, mu_1)
  cluster <- rep(0, length(x))
  cluster[T_0 < .5] <- 1 # assign i to cluster 0 if T_0i > .5

  # measure of accuracy
  cluster_true <- c(rep(0, n), rep(1, n)) # true clusters
  accuracy <- sum(cluster == cluster_true) / (2 * n)
  result <- c(result, accuracy)
  return(result)
}

p7_1 <- EM2(50, 0.1, 0.9)
p7_2 <- EM2(50, 0.5, 0.95)
p7_3 <- EM2(50, 0.25, 0.6)
p7_4 <- EM2(50, 0.7, 0.1)
p7_5 <- EM2(50, 0.5, 0.5)
acc_7 <- c(p7_1[4], p7_2[4], p7_3[4], p7_4[4], p7_5[4])
barplot(acc_7, names.arg=c("(0.1, 0.9)", "(0.5, 0.95)", "(0.25, 0.6)", "(0.7, 0.1)", "(0.5, 0.5)"), xlal
```



Similarly to previous results, this bar chart shows that the accuracy of poisson EM clustering with binomial distributions is very high. As expected, accuracy drops to 50% when the mixing distributions are identical (it's like choosing 50/50 every time).