

# S13 - Calsificacion no supervizada

Juan Carlos Martinez-Ovando ITAM

## Simulacion de datos

Generamos una muestra de  $n$  variables escalaras de una mezcla de  $K$  distribuciones gaussianas  $d$  dimensionales. En el siguiente script,  $k$  es el numero de grupos o clases,  $\mu$  es una matriz de dimension  $k \times d$  con vectores de medias para cada grupo, y  $\Sigma$  es un arreglo de dimension  $K \times d \times d$  de matrices de covarianza.

```
rmixgaussian <- function(n, k, mu, sig) {  
  if(!require('MASS')){install.packages("MASS")}  
  library("MASS")  
  
  d <- length(mu[1,])  
  result <- matrix(rep(NA,n*d), ncol=d)  
  colnames(result) <- paste0("X",1:d)  
  
  for(i in 1:n) {  
    result[i,] <- mvrnorm(1, mu = mu[k[i],], Sigma=sig[, ,k[i]])  
  }  
  return(result)  
}
```

Los datos simulados son:

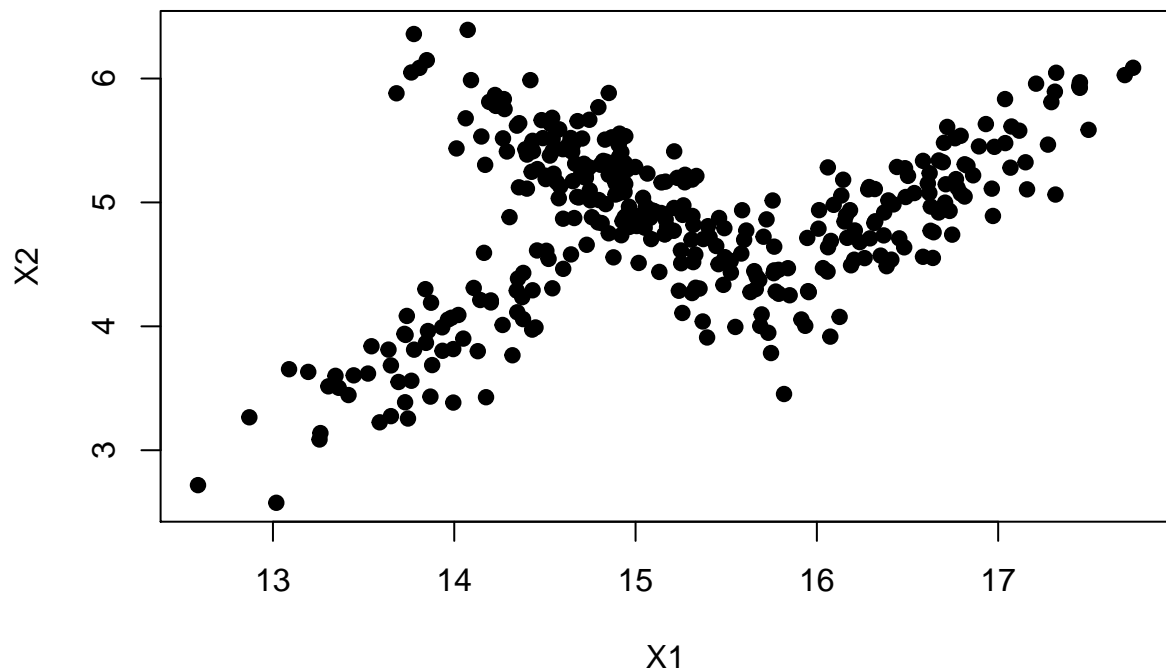
```
set.seed(123)  
  
n <- 360  
  
mu <- matrix(c(14.0,4.0,  
              15.0,5.0,  
              16.5,5.0), ncol=2, byrow=T)  
  
sigs <- array(rep(NA,2*2*3), c(2,2,3))  
  
sigs[, ,1] <- matrix(c(.25, .21, .21,.25), nrow=2, byrow=TRUE)  
sigs[, ,2] <- matrix(c(.25, -.21, -.21,.25), nrow=2, byrow=TRUE)  
sigs[, ,3] <- matrix(c(.25, .21, .21,.25), nrow=2, byrow=TRUE)  
  
pi <- c(.2, .5, .3)  
  
classes <- sample(1:3, n, replace=TRUE, prob=pi)  
  
data <- rmixgaussian(n, classes, mu, sigs)
```

## Loading required package: MASS

### Datos

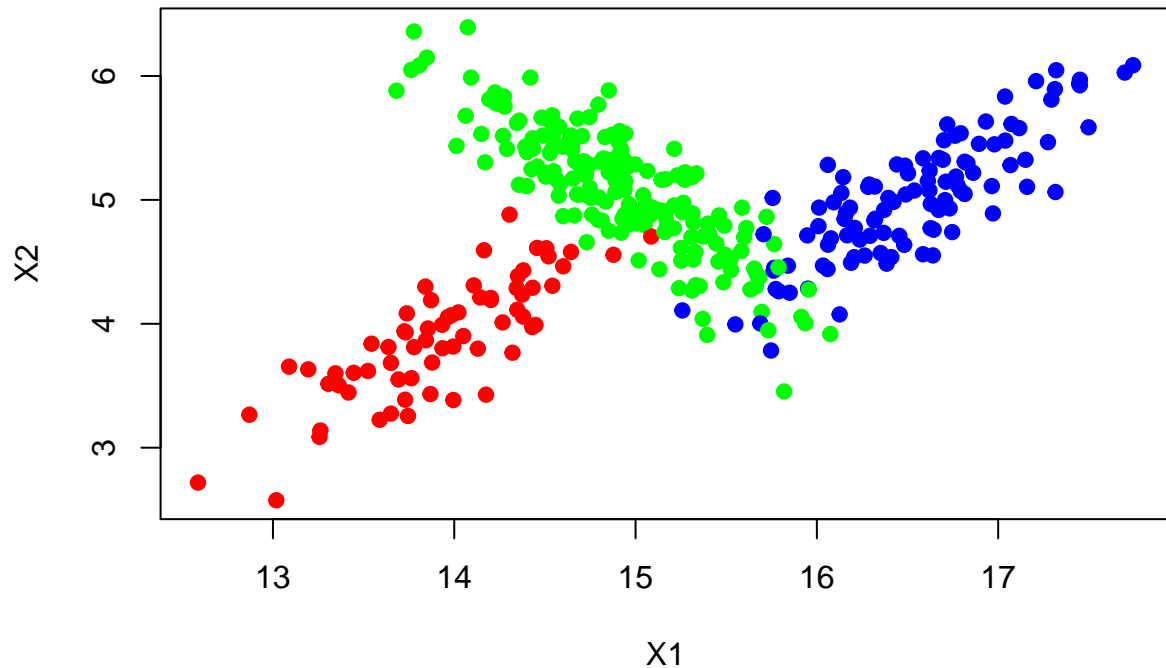
Pretendiendo no conocer con anticipacion como los datos son generado o, inclusive, si estos estan segmentados, su representacion grafica seria de la siguiente forma.

```
plot(data, col="black", xlab="X1", ylab="X2", pch=19)
```



Aunque sabiendo el mecanismo que genero los datos, la representacion grafica de los mismos deberia de ser de la siguiente forma.

```
plot(data, col=c("red","green","blue")[classes], xlab="X1", ylab="X2", pch=19)
```



Los metodos que revisaremos hoy tiene como proposito revelar estructuras de agrupamiento en datos que no conocemos con antelacion. La primera es basada en la nocion geometrica de distancia (**K-means**) y la segunda basada en argumentos de mezclas de modelos (**mixture of gaussian**).

Notaran que la segunda, en el contexto controlado de esta nota la opcion **mixture of gaussian** puede parecer mas adecuada. Sin embargo, ambos modelos sirven al mismo proposito. En clase comentaremos que aspectos son relevantes para uno u otro.

### K-means

El algoritmo de clasificacion **K-mean** funciona como un algoritmo iterativo de asignacion de clases basado en un conjunto de datos y una distancia (tipicamente la distancia euclidiana). El algoritmo funciona sobre un conjunto de datos  $x = x(x_i)_{i=1}^n$   $d$ -dimensionales, y un numero fijo de clases o grupos  $K$ .

Cada grupo  $k$  es caracterizado por un centroide,  $m_k$  definido itertivamente como la media aritmetica de los datos que componen el grupo  $k$ , siendo el grupo  $k$  definido como el conjunto de datos que minimiza la suma de cuadrados de las distancias de los datos a los centroides correspondientes.

El algoritmo se puede expresar como un problema de decision empleando la funcion objetivo

$$J(x, K, \mu) = \sum_{i=1}^n \sum_{k=1}^K r_{ik} d(x_i, \mu_k),$$

donde

- $r_i = (r_{i1}, \dots, r_{iK})$  es la variable indicadora  $K$ -dimensional para la que solo una entrada es igual a 1 (la entrada correspondiente al grupo en el que la observación  $i$  es asignada a la clase  $k$ )
- $\mu = (\mu_i)_{i=1}^K$  es la colección de centroides de las  $K$  clases
- $d(x_i, \mu_k)$  es la distancia (euclidiana) del punto  $x_i$  al centroide  $\mu_k$ .

### Asignación

La asignación en este modelo consiste en minimizar la función  $J(x, K)$  para  $r = (r_i)_{i=1}^n$ , dado los centroides  $(\mu_k)_{k=1}^K$ . Y con base en las asignaciones  $r$  minimizar  $J(x, K, \mu)$  con respecto a  $\mu$ .

### Algoritmo

El algoritmo de este procedimiento se muestra a continuación:

```
k.means <- function(dataset, K, max_iter=100) {

  get_classes <- function(rnk){
    apply(rnk, 1, function(row) which.max(row))
  }

  d <- ncol(dataset)
  N <- nrow(dataset)
  ranges <- sapply(1:d, function(i) range(dataset[,i]))

  # K centroides iniciales
  mu <- t(replicate(K, sapply(1:d,
                             function(i) runif(1, ranges[1,i], ranges[2,i]))))

  rnk <- matrix(rep(0, K*n), ncol=K)
  old_classes <- get_classes(rnk)

  for(it in 1:max_iter) {
    for(n in 1:N) {
      distances <- sapply(1:K, function(k)
        norm(as.matrix(dataset[n,] - mu[k,]), "F"))
      rnk[n,] <- rep(0, K)
      rnk[n, which.min(distances)] <- 1
    }

    classes <- get_classes(rnk)
    if (all(old_classes == classes))
      break
    else
      old_classes <- classes

    for(k in 1:K) {
      mu[k,] <- rnk[,k] %*% dataset / sum(rnk[,k])
    }
  }
}
```

```

    }
  }
  output <- list(mu=mu, pred=classes)
  return(output)
}

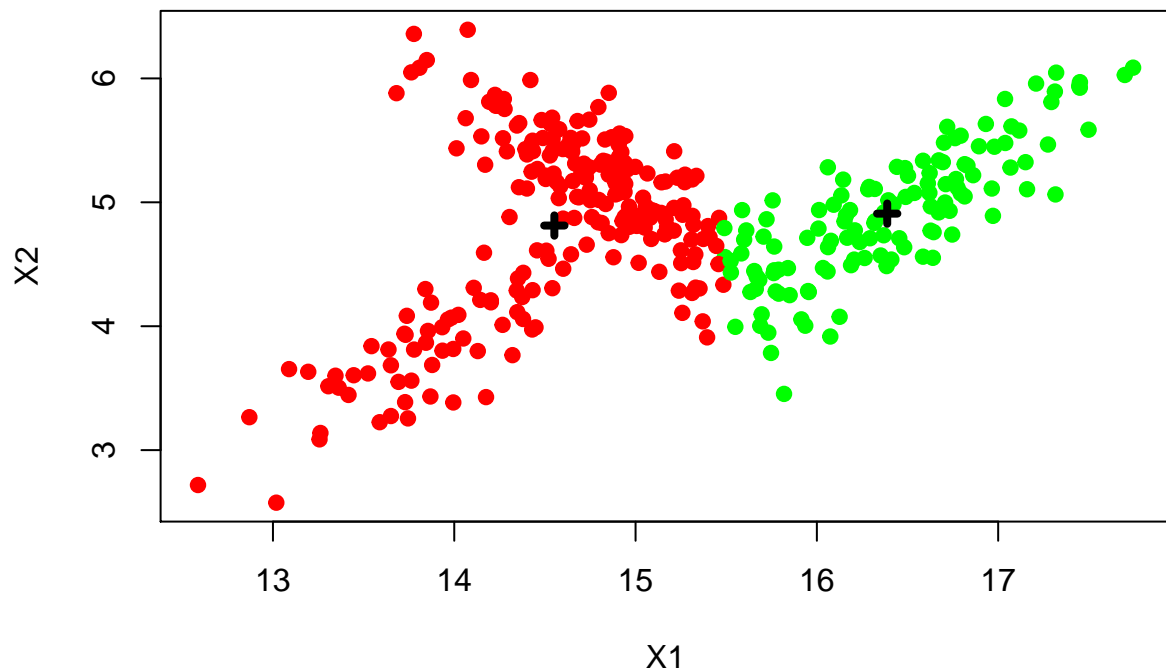
```

```

set.seed(1231)
result <- k.means(data,2)

plot(data,
      col=c("red","green","blue")[result$pred],
      xlab="X1", ylab="X2", pch=19)
points(result$mu, pch=3, lwd=4)

```

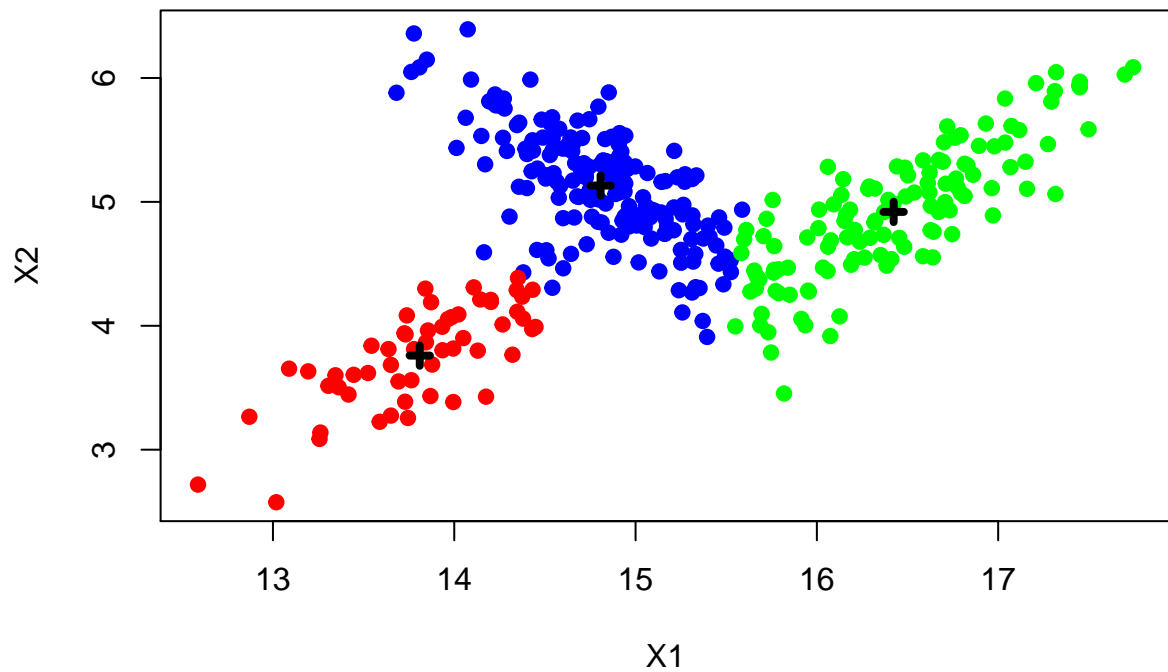


```

set.seed(1231)
result <- k.means(data,3)

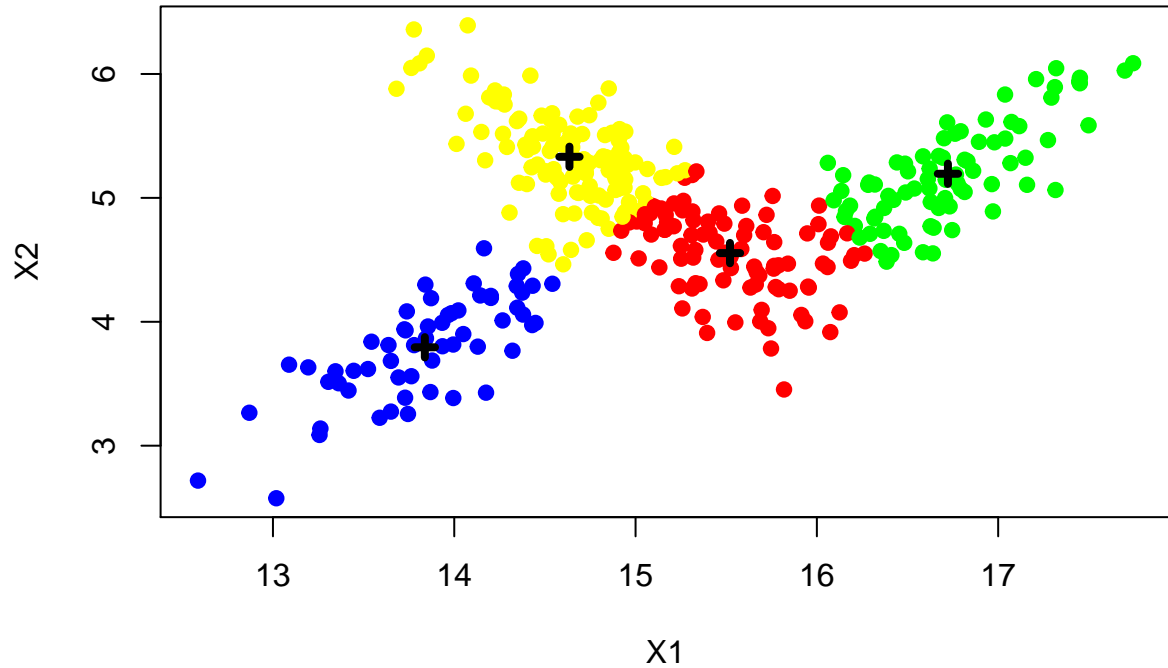
plot(data,
      col=c("red","green","blue")[result$pred],
      xlab="X1", ylab="X2", pch=19)
points(result$mu, pch=3, lwd=4)

```



```
set.seed(1231)
result <- k.means(data,4)

plot(data,
      col=c("red","green","blue","yellow")[result$pred],
      xlab="X1", ylab="X2", pch=19)
points(result$mu, pch=3, lwd=4)
```



## Mixture of Gaussian

El modelo que describiremos a contiucion realiza *clasificacion no supervisada* de datos empleando argumentos no probabilisticos (no distancias). Lo vimos descrito brevemente en la sesion anterior. Ahora lo revisaremos en detalle.

### Modelo

Consideramos que los datos  $(x_i)_{i=1}^n$  toman valores en  $\mathbb{R}^p$  (i.e. este es el soporte del modelo). El desconocimiento (*aleatoriedad implicita*) sobre los datos se describe mediante la siguiente densidad

$$p(x) = \sum_{k=1}^K p(Gpo_k) p(x|Gpo_k) \quad (1)$$

$$= \sum_k w_k N(x|\mu_k, \Sigma_k), \quad (2)$$

donde

- $K$  es el numero implicito de grupos,
- $w_k$  es la probabilidad de que el dato  $x$  sea descrito (pertenezca) por el grupo  $k$ , para  $k = 1, \dots, K$

- $N(x|\mu_k, \Sigma_k)$  es la distribucion que describe a la dispersion subyacente dentro del grupo  $k$ , a traves de los parametros implicitos  $(\mu_k, \Sigma_k)$ , para  $k = 1, \dots, K$ .

En la descripcion anterior,  $\mu_k$  se interpreta similarmente a los *centroides* del modelo **K-means**, mientras que  $\Sigma_k$  describe que tan dispersos pueden ser los datos al rededor de  $\Sigma_k$ .

**Nota:** En esta especificacion se considera que  $K$  (numero total de grupos) es conocido o fijo. Sin embargo, el modelo puede extenderse para incluir incertidumbre sobre  $K$  tambien.

Los parametros del modelo, entonoces, son  $(w, \mu, \Sigma) = (w_k, \mu_k, \Sigma_k)_{k=1}^K$ , con la restriccion  $w_k > 0$ , para  $k = 1, \dots, K$ , y  $\sum_{k=1}^K w_k = 1$ .

### Asignacion

Como mencionamos antes, podemos *modificar*  $w_k$  con base en la evidencia de cada dato particular (o en grupo), a traves de la probabilidad condicional  $w_k(x) = p(Gpo_k|x)$ , que se conoce como probabilidad de asignacion, la cual incorpora la *verosimilitud* de que  $x$  sea descrito por  $N(x|\mu_k, \Sigma_k)$ , i.e.

$$w_k(x) = \frac{w_k N(x|\mu_k, \Sigma_k)}{\sum_{l=1}^K w_l N(x|\mu_l, \Sigma_l)},$$

para  $k = 1, \dots, K$ .

Asi, el dato  $x$  es asignado a la clase  $k^*$  que cumpla

$$k^* = \operatorname{argmax}_k \{w_k(x) : k = 1, \dots, K\}.$$

### Aprendizaje estadistico

Podemos estimar (inferir) los valores de los parametros  $(w, \mu, \Sigma)$  empleando la verosimilitud

$$p(w, \mu, \Sigma|x) = \prod_{i=1}^n \sum_{k=1}^K w_k N(x_i|\mu_k, \Sigma_k),$$

para  $x = (x_i)_{i=1}^n$ . Esta verosimilitud, como comentamos, es intratable analiticamente.

Los enfoques bayesianos y frecuentistas de inferencia sufren por este aspecto del modelo. Sin embargo, esta complejidad se resuelve empleando metodos numericos.

A continuacion revisaremos un enfoque frecuentista basado en el algoritmo EM (Dempster, et al, 1989).

### Algoritmo EM

El algoritmo EM para mezclas (en general) consiste en incluir variables de asignacion latentes para cada dato,  $(z_i)_{i=1}^n$ , donde cada  $z_i$  es un vector binario  $K$  dimensional,  $z_i = (z_{i1}, \dots, z_{iK})$ , con

$$z_{ik} = \begin{cases} 1, & x_i \in Gpo_k \\ 0, & \text{e.o.c.} \end{cases}$$

Los parametros  $w$  representan la distribucion marginal de las  $z_i$ s, de forma que la *pseudo-verosimilitud* para  $z = (z_i)_{i=1}^n$  esta dada por

$$p(z|w) = \prod_{k=1}^K w_k^{n_k},$$



donde

$$n_k = \#\{z_i : z_{ik} = 1\}.$$

La distribucion implicita de las  $z_i$ s es obviamente multinomial.

Ahora, incorporando la evidencia contenida en los datos,  $x_i$ s, la verosimilitud de  $z$  en  $x$  (un solo dato) se describe por

$$p(x_i|z_i) = \prod_{k=1}^K N(x_i|\mu_k, \Sigma_k)^{z_{ik}} = N(x_i|\mu_k, \Sigma_k),$$

para  $k$  tal que  $z_{ik} = 1$ .

La verosimilitud extendida para  $(w, \mu, \Sigma, z)$  ddo  $x$  es definida ahora como

$$p(x, z|w, \mu, \Sigma) \propto \prod_{i=1}^n \prod_{k=1}^K w_k^{z_{ik}} N(x_i|\mu_k, \Sigma_k)^{z_{ik}},$$

la cual esta definida para todos los  $z_{ik}$ s iguales a 1.

El algoritmo EM descansa en dos pasos iterativos, de esperanza (E) y de optimizacion (M).

*Paso E*

En cada iteracion se calcula el valor esperado de las variables latentes  $z_i$ s, con base en

$$E(z|x, w, \mu, \Sigma) [\log p(x, z|w, \mu, \Sigma)] = \sum_{i=1}^n \sum_{k=1}^K w_k(x_i) (w_k + \log N(x_i|\mu_k, \Sigma_k)),$$

con  $w_k(x_i)$  definida como antes.

*Paso M*

El paso M consiste en maximizar  $E(z|x, w, \mu, \Sigma) [\log p(x, z|w, \mu, \Sigma)]$  con respecto a  $(w, \mu, \Sigma)$ , dados  $x$  y  $z$ .

Para este modelo, los valores de  $(w, \mu, \Sigma)$  que maximizan la expresion referida tienen una expresion analitica cerrada, dada por

$$\mu_k = \frac{1}{n_k} \sum_{i=1}^n w_k(x_i) x_i \quad (3)$$

$$\Sigma_k = \frac{1}{n_k} \sum_{i=1}^n w_k(x_i) (x_i - \mu_k)' (x_i - \mu_k) \quad (4)$$

$$w_k = \frac{n_k}{n}, \quad (5)$$

para  $k = 1, \dots, K$ , con  $n_k$  definida como antes.

La implementacion de este algoritmo se ilustra a continuacion:

```
mixgaussian.em <- function(dataset, K, max_iter=100, epsilon=1e-3) {
  if(!require('mvtnorm')){install.packages("mvtnorm")}
  library("mvtnorm")
}
```

```

get_classes <- function(gammak){
  apply(gammak,1,function(row) which.max(row))
}

d <- ncol(dataset)
N <- nrow(dataset)
ranges <- sapply(1:d, function (i) range(dataset[,i]))

# Inicializacion
pik <- rep(1/K,K)
muk <- t(replicate(K,sapply(1:d,
                           function(i) runif(1,ranges[1,i], ranges[2,i]))))
Sigmas <- array(rep(NA,2*2*3), c(2,2,3))
for (k in 1:K){
  Sigmas[, ,k] <- diag(d)
}
gammak <- matrix(rep(0,K*N),ncol=K)
old_gammak <- gammak

# Pasos EM
for(it in 1:max_iter){
  # Paso M
  for (k in 1:K){
    gammak[,k] <- apply(dataset, 1,
                      function(xi) {
                        pik[k] * dmnorm(xi,muk[k,], Sigmas[, ,k])
                      })
  }
  gammak <- t(apply(gammak, 1, function(row) row/sum(row)))

  if (sum(abs(gammak - old_gammak)) < epsilon)
    break
  else
    old_gammak <- gammak

  # Paso M
  Nk <- sapply(1:K, function (k) sum(gammak[,k]))
  pik <- Nk/N
  for(k in 1:K){
    muk[k,] <- apply(gammak[,k] * dataset,2,sum) / Nk[k]
    Sigmas[, ,k] <- diag(d) * 0
    for(n in 1:N){
      Sigmas[, ,k] <- Sigmas[, ,k] +
        gammak[n,k]*
        (dataset[n,]-muk[k,])%*%t(dataset[n,]-muk[k,])
    }
    Sigmas[, ,k] <- Sigmas[, ,k] / Nk[k]
  }
}

```

```

    }
  }
  output <- list(mu=mu, Sigmas=Sigmas, gammak=gammak, pred=get_classes(gammak))
  return(output)
}

```

```

set.seed(101)
result <- mixgaussian.em(data,2)

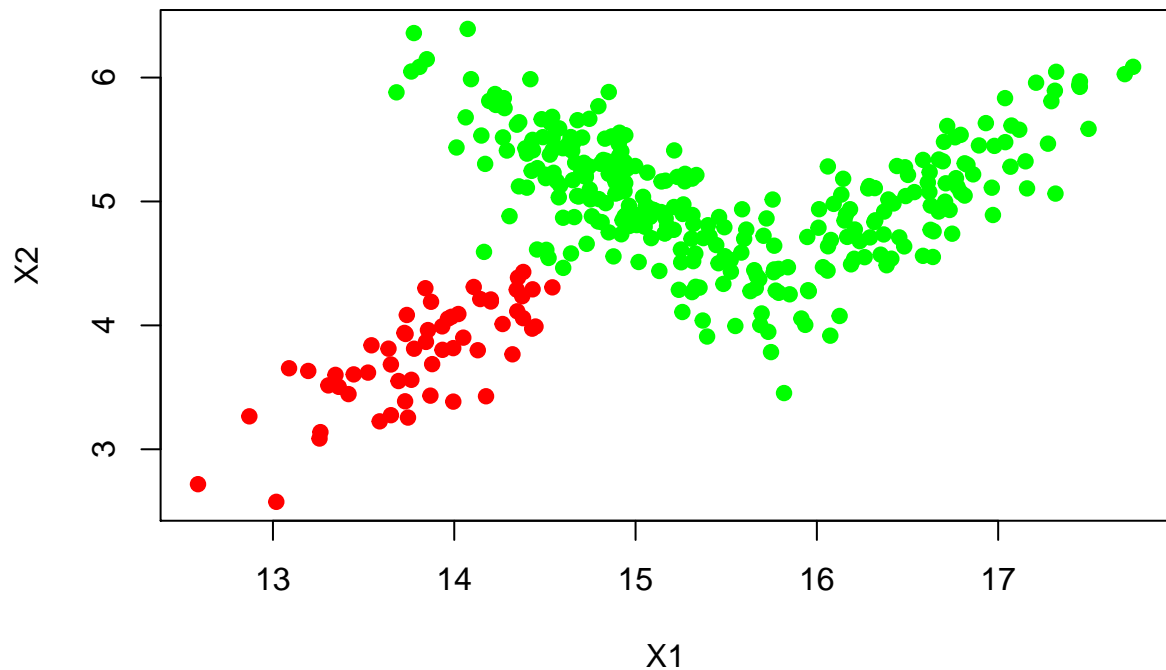
```

## Loading required package: mvtnorm

```

plot(data, col=c("red","green")[result$pred],
      xlab="X1", ylab="X2", pch=19)

```

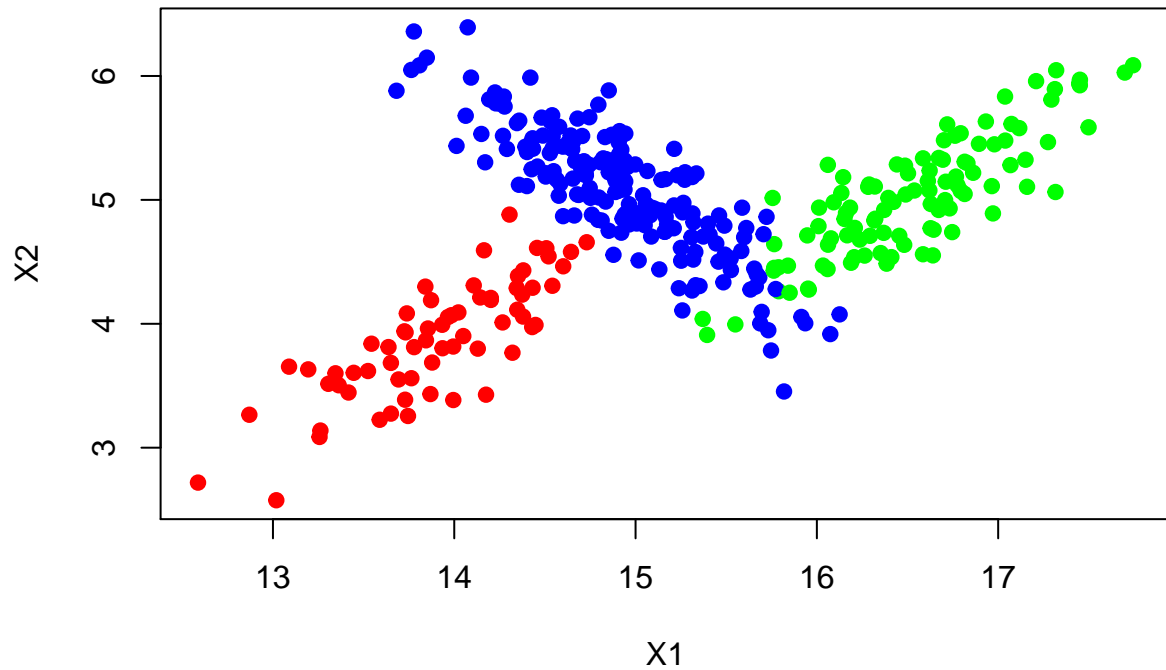


```

set.seed(101)
result <- mixgaussian.em(data,3)

plot(data, col=c("red","green","blue")[result$pred],
      xlab="X1", ylab="X2", pch=19)

```



```
{r mixgaussian_ex4}  
set.seed(101)  
result <- mixgaussian.em(data,4)  
  
plot(data, col=c("red","green","blue","yellow")[result$pred],  
      xlab="X1", ylab="X2", pch=19)
```

---

### Referencias adicionales

- **Bishop**, *Pattern Recognition and Machine Learning* (Book)