

Práctica 1 Aprendizaje Automático

Jacinto Carrasco Castillo

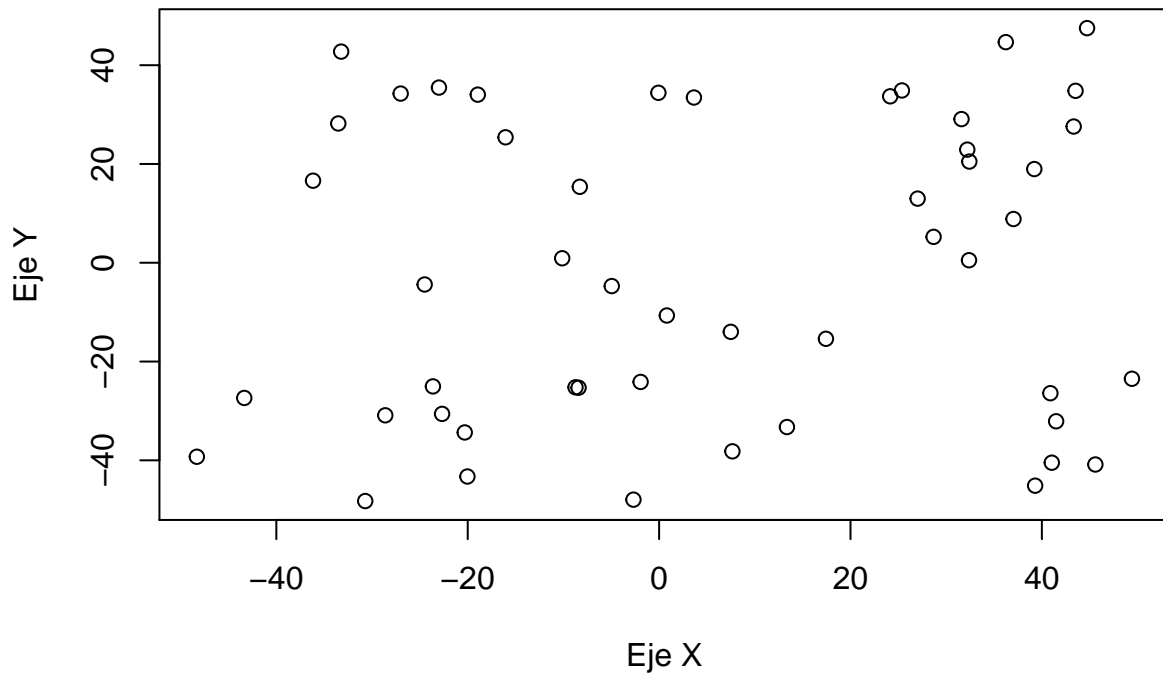
Generación y visualización de datos

```
set.seed(3141592)
simula_unif <- function(N, dim, rango){
  lista <- matrix(runif(N*dim, min = rango[1], max = rango[2]), dim, N)
  return(lista)
}
```

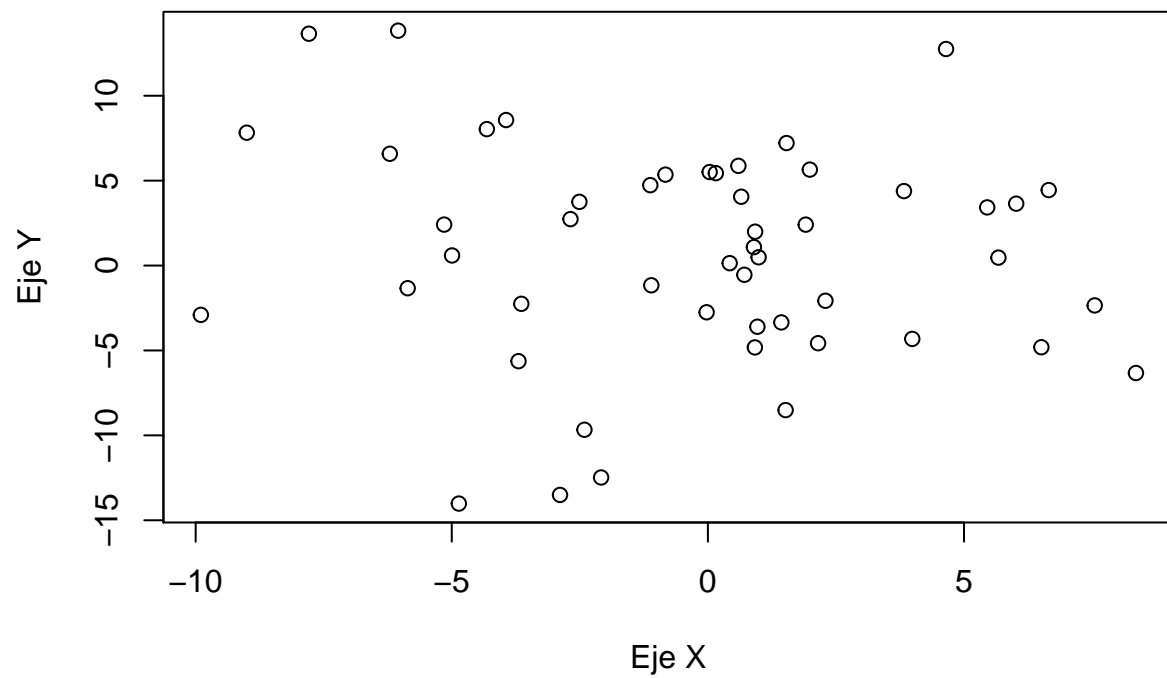
```
simula_gauss <- function(N, dim, sigma){
  lista <- matrix(rnorm(N*dim, sd = sigma), dim, N)
  return(lista)
}
```

```
lista_unif <- simula_unif(50, 2, c(-50, 50))
lista_gauss <- simula_gauss(50, 2, c(5,7))
```

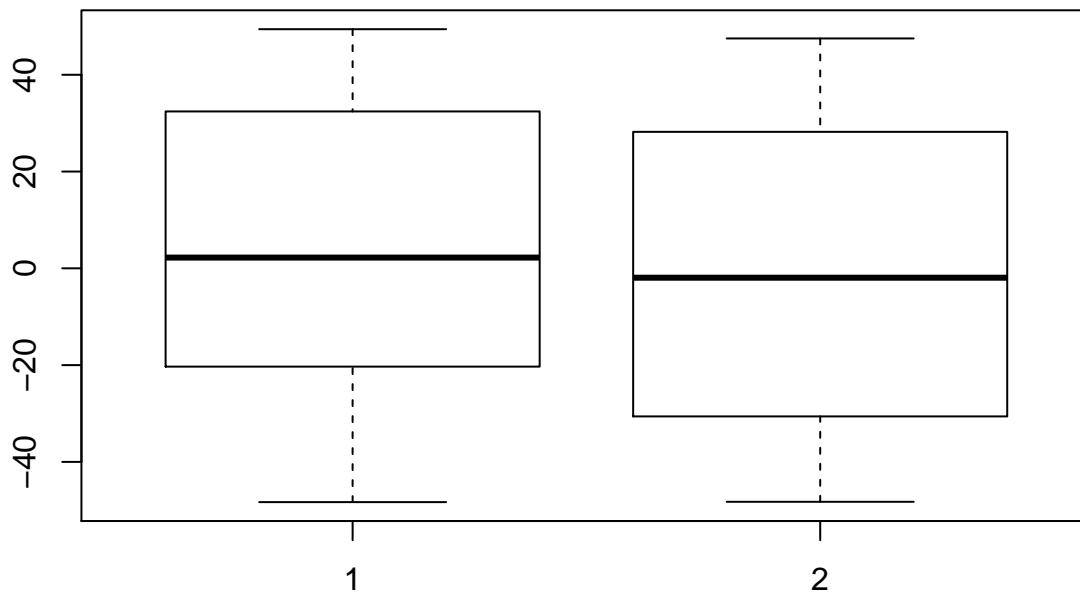
```
plot(lista_unif[1,], lista_unif[2,], xlab="Eje X", ylab="Eje Y")
```



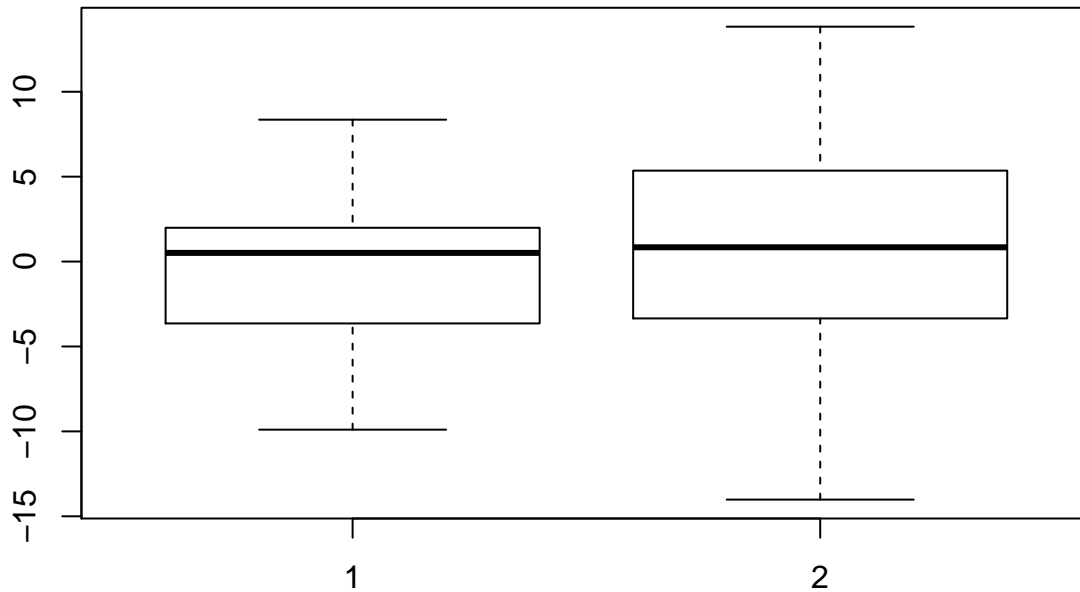
```
plot(lista_gauss[1,], lista_gauss[2,], xlab="Eje X", ylab="Eje Y")
```



```
boxplot(lista_unif[1,],lista_unif[2,])
```



```
boxplot(lista_gauss[1,],lista_gauss[2,])
```



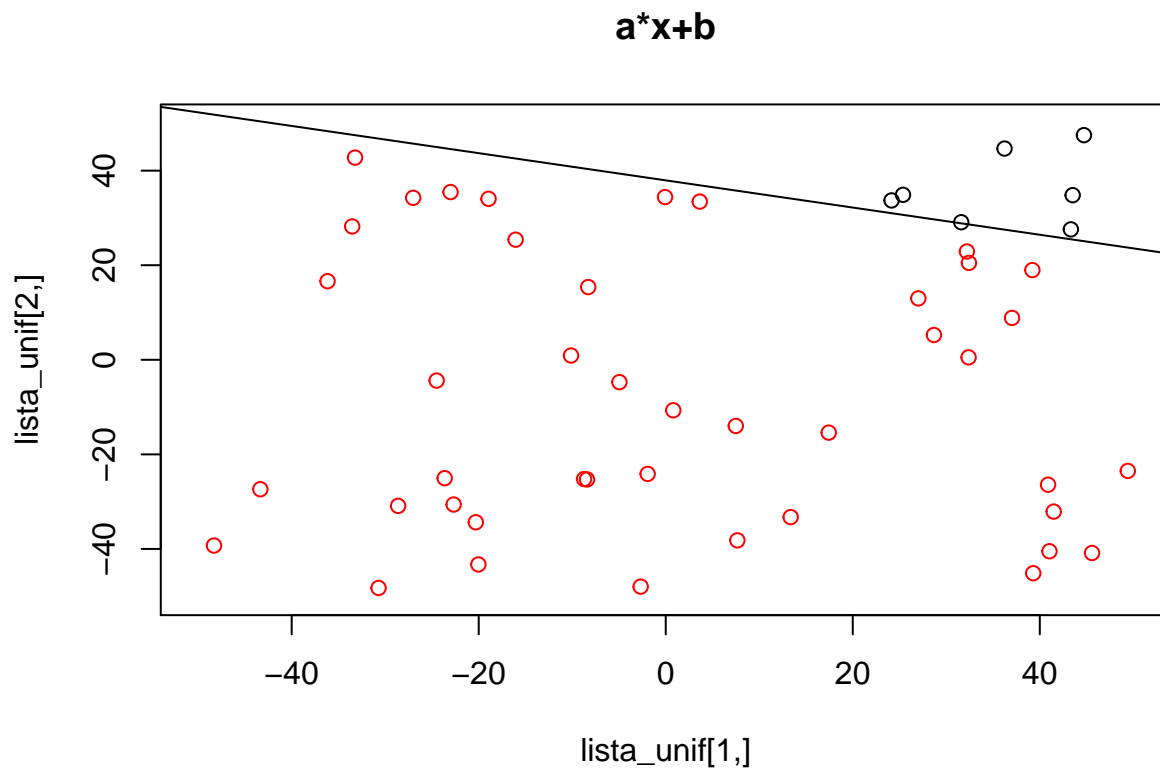
```
simula_recta <- function(intervalo){
  puntos <- simula_unif(2,2,intervalo)
  a <- (puntos[2,2]-puntos[2,1])/(puntos[1,2]-puntos[1,1])
  b <- puntos[2,1] - a * puntos[1,1]
  return(c(a,b))
}
```

```
intervalo = c(-50, 50)
coefs <- simula_recta(intervalo)
a <- coefs[1]
b <- coefs[2]

f <- function(X){
  return(X[2] - a*X[1] -b)
}

etiqueta = apply(lista_unif, 2, function(X) sign(f(X)))

plot.new()
plot.window(xlim=c(-50,50), ylim=c(-50,50))
axis(1)
axis(2)
box()
title(main="a*x+b", ylab = "lista_unif[2,]", xlab = "lista_unif[1,]")
points(lista_unif[1,etiqueta==1], lista_unif[2,etiqueta==1], col = 2)
points(lista_unif[1,etiqueta==1], lista_unif[2,etiqueta==1])
abline(b,a)
```



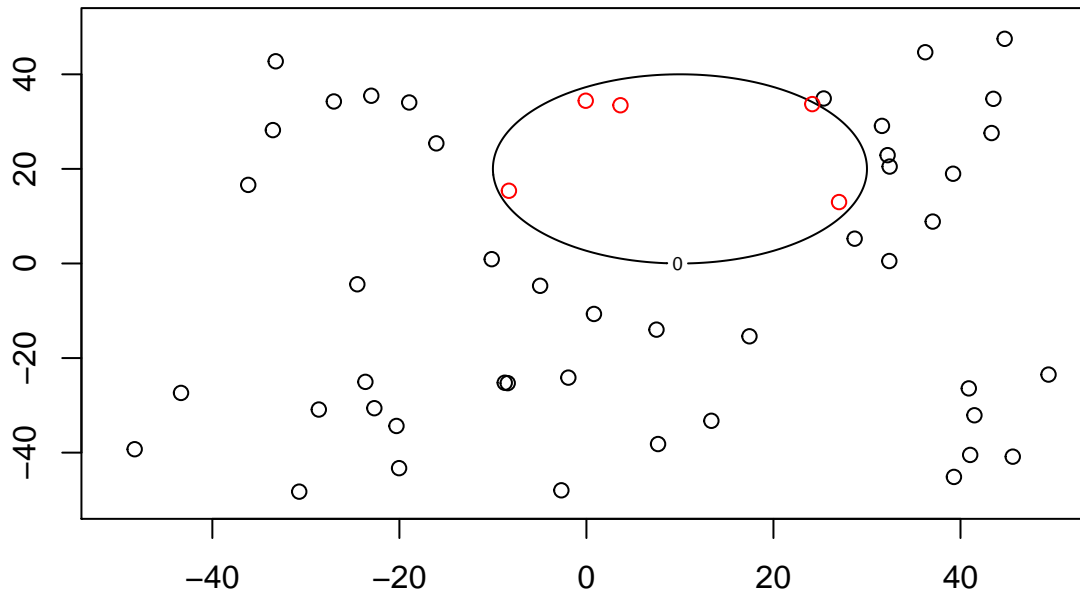
```
f_1 <- function(X){
  return((X[1] - 10)^2 + (X[2] - 20)^2 - 400)
}

etiqueta_1 = apply(lista_unif, 2, function(X) sign(f_1(X)))

x <- seq(-50,50,length=1000)
y <- seq(-50,50,length=1000)
z <- outer(x,y,function(x,y) (x - 10)^2 + (y - 20)^2 - 400)

contour(x,y,z, levels=0, main=expression((x-10)^2 + (y-20)^2 - 400))
points(lista_unif[1,etiqueta_1==1], lista_unif[2,etiqueta_1==1])
points(lista_unif[1,etiqueta_1==-1], lista_unif[2,etiqueta_1==-1],col=2)
```

$$(x-10)^2 + (y-20)^2 - 400$$



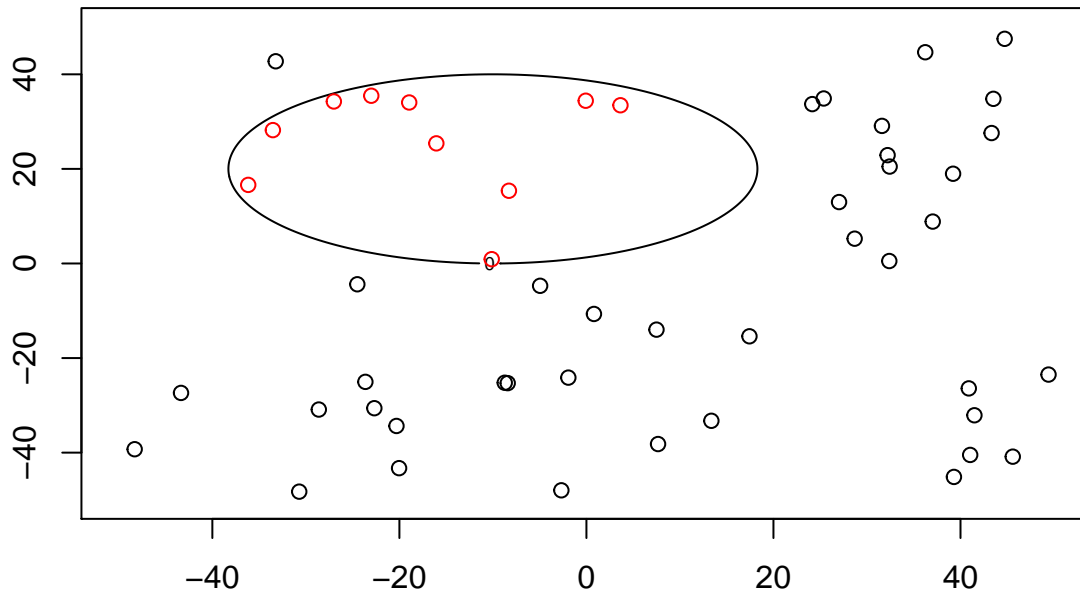
```
f_2 <- function(X){
  return(0.5*(X[1] + 10)^2 + (X[2] - 20)^2 - 400)
}

etiqueta_2 = apply(lista_unif, 2, function(X) sign(f_2(X)))

x <- seq(-50,50,length=1000)
y <- seq(-50,50,length=1000)
z <- outer(x,y,function(x,y) 0.5*(x + 10)^2 + (y - 20)^2 - 400)

contour(x,y,z, levels=0, main=expression(0.5(x + 10)^2 + (y - 20)^2 - 400))
points(lista_unif[1,etiqueta_2==1], lista_unif[2,etiqueta_2==1])
points(lista_unif[1,etiqueta_2==-1], lista_unif[2,etiqueta_2==-1],col=2)
```

$$0.5(x+10)^2 + (y-20)^2 - 400$$



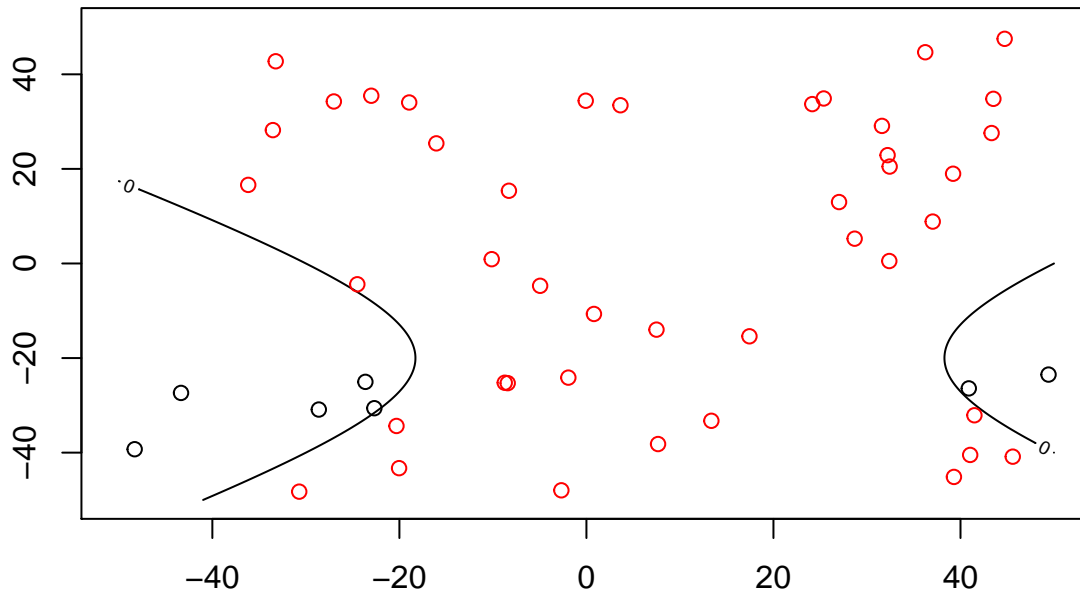
```
f_3 <- function(X){
  return(0.5*(X[1] - 10)^2 - (X[2] + 20)^2 - 400)
}

etiqueta_3 = apply(lista_unif, 2, function(X) sign(f_3(X)))

x <- seq(-50,50,length=1000)
y <- seq(-50,50,length=1000)
z <- outer(x,y,function(x,y) 0.5*(x - 10)^2 - (y + 20)^2 - 400)

contour(x,y,z, levels=0, main=expression( 0.5*(x - 10)^2 - (y + 20)^2 - 400))
points(lista_unif[1,etiqueta_3==1], lista_unif[2,etiqueta_3==1])
points(lista_unif[1,etiqueta_3==-1], lista_unif[2,etiqueta_3==-1],col=2)
```

$$0.5(x-10)^2 - (y+20)^2 - 400$$



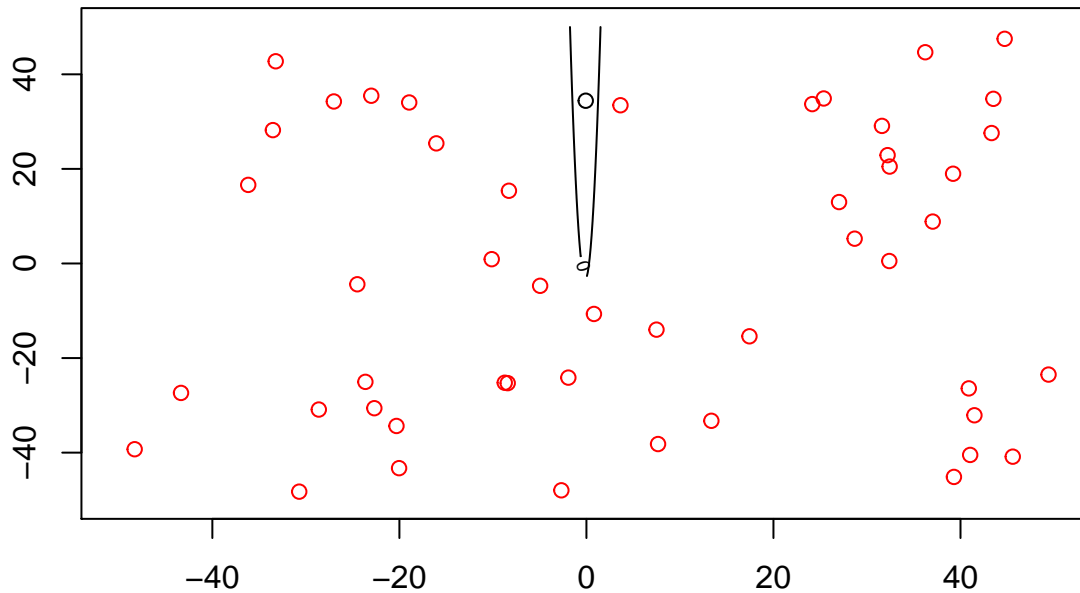
```
f_4 <- function(X){
  return(X[2] - 20*X[1]^2 - 5*X[1] + 3)
}

etiqueta_4 = apply(lista_unif, 2, function(X) sign(f_4(X)))

x <- seq(-50,50,length=1000)
y <- seq(-50,50,length=1000)
z <- outer(x,y,function(x,y) y - 20*x^2 - 5*x + 3)

contour(x,y,z, levels=0, main=expression(y - 20*x^2 - 5*x + 3))
points(lista_unif[1,etiqueta_4==1], lista_unif[2,etiqueta_4==1])
points(lista_unif[1,etiqueta_4==-1], lista_unif[2,etiqueta_4==-1],col=2)
```

$$y - 20x^2 - 5x + 3$$



```
modify_rnd_bool_subvector <- function(v, perc = 0.1){
  mod_v <- v

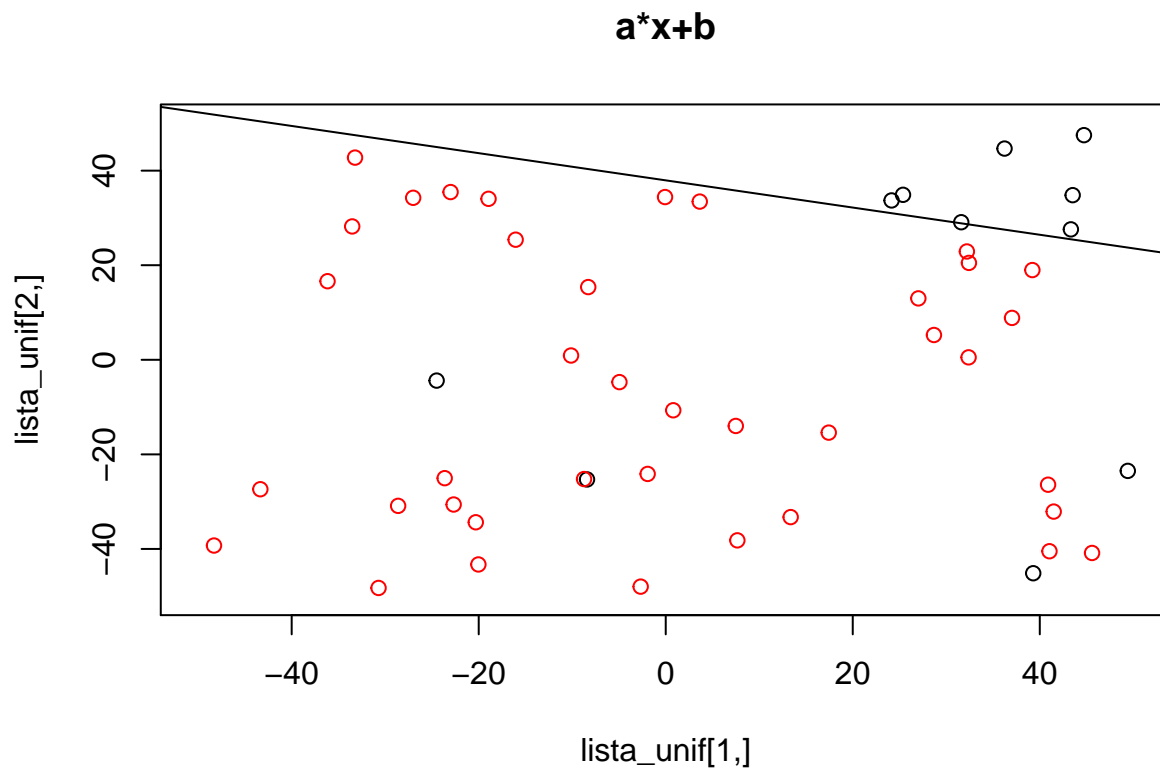
  if( length(which(v == 1)) >= 10){
    to_change <- sample(which(v == 1), 0.1*length(which(v == 1)) )
    mod_v[to_change] = -1
  }

  if( length(which(v == -1)) >= 10){
    to_change <- sample(which(v == -1), 0.1*length(which(v == -1)) )
    mod_v[to_change] = 1
  }

  return(mod_v)
}
```

```
etiqueta_mod <- modify_rnd_bool_subvector(etiqueta)

plot.new()
plot.window(xlim=c(-50,50), ylim=c(-50,50))
axis(1)
axis(2)
box()
title(main="a*x+b", ylab = "lista_unif[2,]", xlab = "lista_unif[1,]")
points(lista_unif[1,etiqueta_mod==1], lista_unif[2,etiqueta_mod==1])
points(lista_unif[1,etiqueta_mod== -1], lista_unif[2,etiqueta_mod== -1], col=2)
abline(b,a)
```

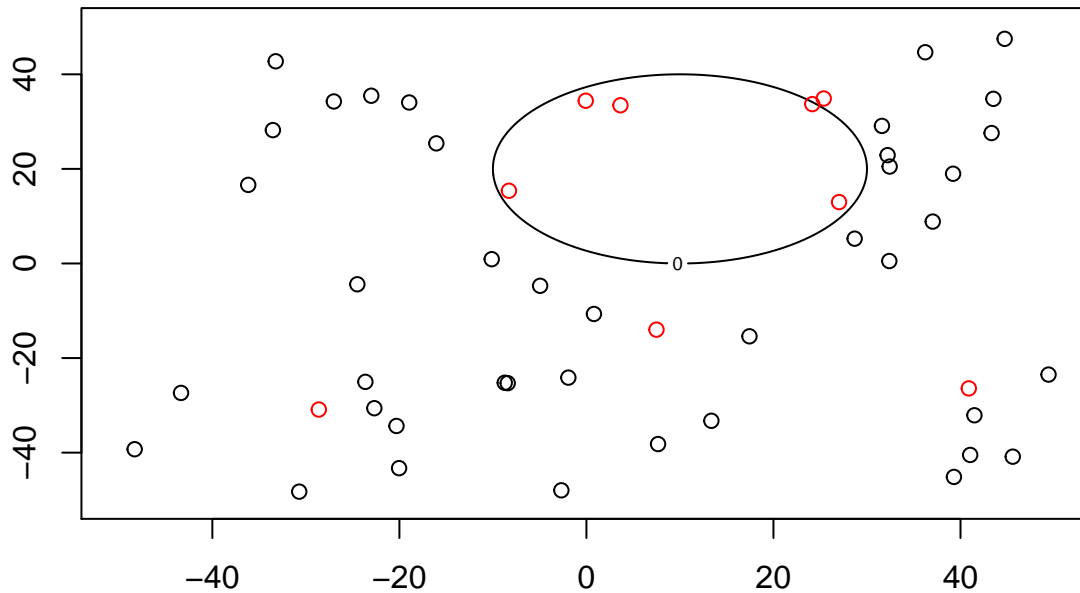



```
etiqueta_mod_1 <- modify_rnd_bool_subvector(etiqueta_1)

x <- seq(-50,50,length=1000)
y <- seq(-50,50,length=1000)
z <- outer(x,y,function(x,y) (x - 10)^2 + (y - 20)^2 - 400)

contour(x,y,z, levels=0, main=expression((x-10)^2 + (y-20)^2 - 400))
points(lista_unif[1,etiqueta_mod_1==1], lista_unif[2,etiqueta_mod_1==1])
points(lista_unif[1,etiqueta_mod_1==1], lista_unif[2,etiqueta_mod_1==1],col=2)
```

$$(x-10)^2 + (y-20)^2 - 400$$

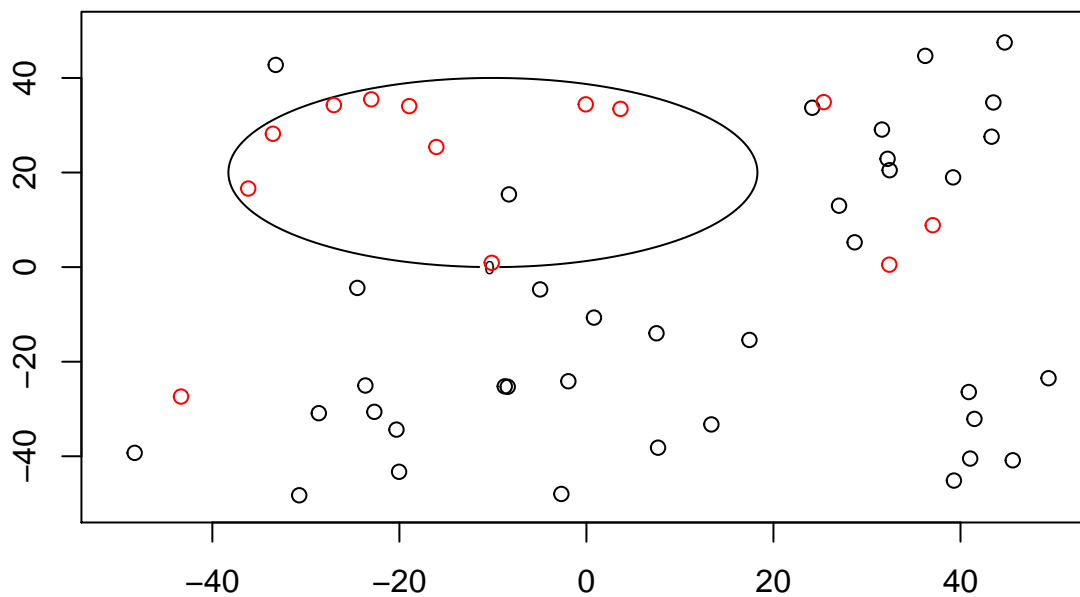


```
etiqueta_mod_2 <- modify_rnd_bool_subvector(etiqueta_2)

x <- seq(-50,50,length=1000)
y <- seq(-50,50,length=1000)
z <- outer(x,y,function(x,y) 0.5*(x + 10)^2 + (y - 20)^2 - 400)

contour(x,y,z, levels=0, main=expression(0.5*(x + 10)^2 + (y - 20)^2 - 400))
points(lista_unif[1,etiqueta_mod_2==1], lista_unif[2,etiqueta_mod_2==1])
points(lista_unif[1,etiqueta_mod_2==1], lista_unif[2,etiqueta_mod_2==1],col=2)
```

$$0.5(x+10)^2 + (y-20)^2 - 400$$



```

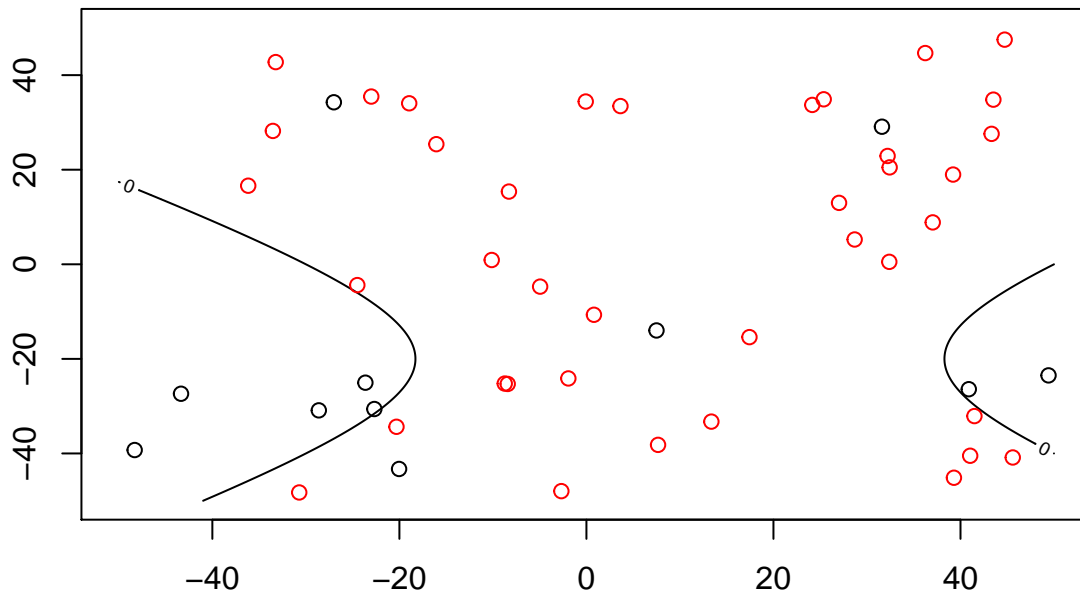
etiqueta_mod_3 <- modify_rnd_bool_subvector(etiqueta_3)

x <- seq(-50,50,length=1000)
y <- seq(-50,50,length=1000)
z <- outer(x,y,function(x,y) 0.5*(x - 10)^2 - (y + 20)^2 - 400)

contour(x,y,z, levels=0, main=expression( 0.5*(x - 10)^2 - (y + 20)^2 - 400))
points(lista_unif[1,etiqueta_mod_3==1], lista_unif[2,etiqueta_mod_3==1])
points(lista_unif[1,etiqueta_mod_3==1], lista_unif[2,etiqueta_mod_3==1],col=2)

```

$$0.5(x - 10)^2 - (y + 20)^2 - 400$$



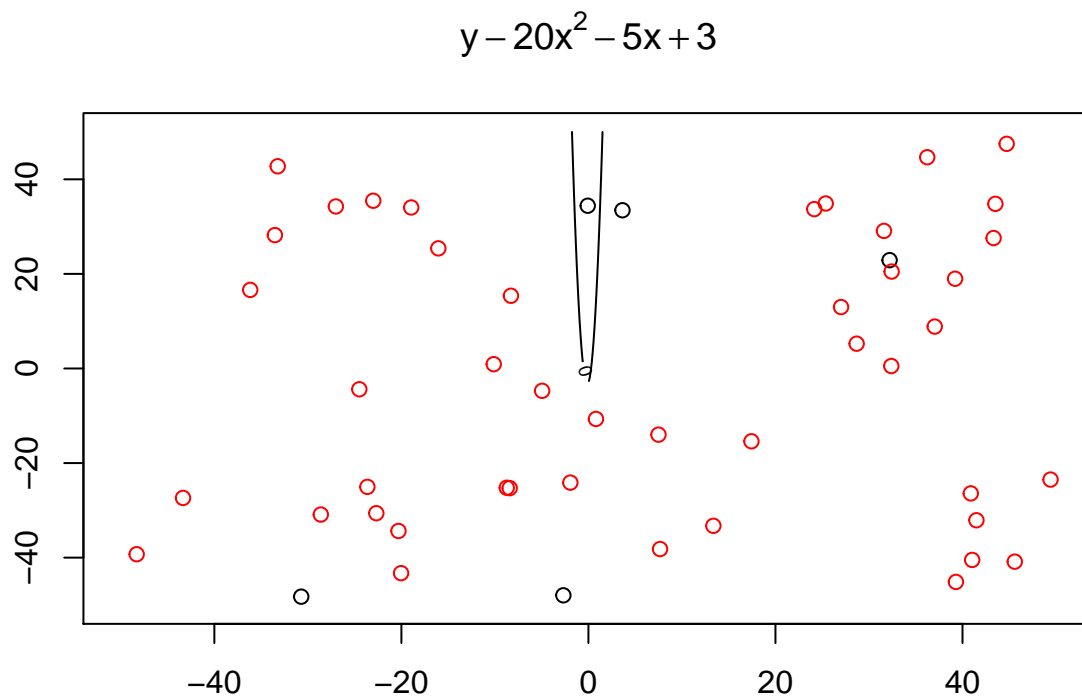
```

etiqueta_mod_4 <- modify_rnd_bool_subvector(etiqueta_4)

x <- seq(-50,50,length=1000)
y <- seq(-50,50,length=1000)
z <- outer(x,y,function(x,y) y - 20*x^2 - 5*x + 3)

contour(x,y,z, levels=0, main=expression(y - 20*x^2 - 5*x + 3))
points(lista_unif[1,etiqueta_mod_4==1], lista_unif[2,etiqueta_mod_4==1])
points(lista_unif[1,etiqueta_mod_4==1], lista_unif[2,etiqueta_mod_4==1],col=2)

```



Ajuste del algoritmo Perceptron

```
ajusta_PLA <- function(datos, label, max_iter, vini, draw_iterations = FALSE){
  sol <- vini
  iter <- 0
  changed <- TRUE

  if(draw_iterations){
    plot.new()
    plot.window(xlim=c(-50,50), ylim=c(-50,50))
    axis(1)
    axis(2)
    box()
    title(main="Iteraciones PLA", ylab = "Y", xlab = "X")
  }

  while ( iter < max_iter && changed){

    changed <- FALSE
    for( inner_iter in 1:ncol(datos) ){
      x <- c(datos[,inner_iter],1)

      if( sign(crossprod(x,sol)) != label[inner_iter]){
        sol <- sol + label[inner_iter] * x
        changed <- TRUE
      }
    }
  }
  if(draw_iterations) abline(-sol[3]/sol[2], -sol[1]/sol[2], col=iter)
  iter <- iter+1
}
```

```

}
sol <- sol/sol[length(sol)-1]

return( list( hiperplane = sol, iterations = iter))
}

sol_pla <- ajusta_PLA(lista_unif,etiqueta, 7000,rep(0,3))$hiperplane

```

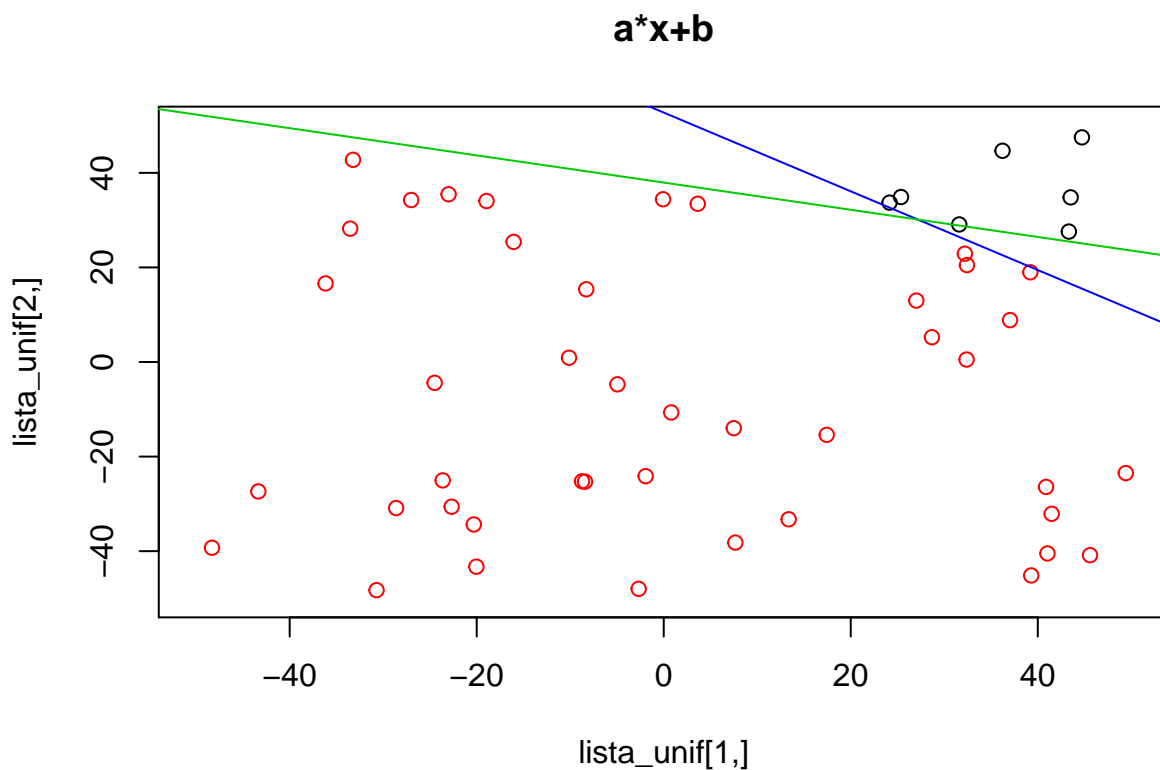
Representación de ejecución del algoritmo PLA

```

plot.new()
plot.window(xlim=c(-50,50), ylim=c(-50,50))
axis(1)
axis(2)
box()
title(main="a*x+b", ylab = "lista_unif[2,]", xlab = "lista_unif[1,]")
points(lista_unif[1,etiqueta==1], lista_unif[2,etiqueta==1], col = 2)
points(lista_unif[1,etiqueta==1], lista_unif[2,etiqueta==1])

abline(-sol_pla[3], -sol_pla[1], col = 4)
abline(b, a, col=3)

```



Cálculo de la media para 10 ejecuciones del PLA con el vector (0,0,0) y vectores aleatorios:

```

iterations <- numeric()
length(iterations) <- 10

iterations[1] <- ajusta_PLA(lista_unif,etiqueta, 10000,rep(0,3))$iterations

```

```
for(i in 2:10){
  v_ini <- runif(3)
  iterations[i] <- ajusta_PLA(lista_unif,etiqueta, 10000, v_ini)$iterations
}
```

```
print(c("Media de iteraciones ", mean(iterations)))
```

```
## [1] "Media de iteraciones " "64.8"
```

```
count_errors <- function(f, datos, label){
  #Conteo de errores
  signes <- apply(datos, 2, function(x) return(sign(f(c(x,1)))))
  v <- signes != label
  return(length(v[v]))
}
```

```
hiperplane_to_function <- function( vec ){
  f <- function(x){
    return( crossprod(vec,x) )
  }

  return(f)
}
```

Conteo de errores

```
sol <- ajusta_PLA(lista_unif,etiqueta_mod, 10,rep(0,3))$hiperplane
count_errors(hiperplane_to_function(sol), lista_unif, etiqueta_mod )
```

```
## [1] 20
```

```
sol <- ajusta_PLA(lista_unif,etiqueta_mod, 100,rep(0,3))$hiperplane
count_errors(hiperplane_to_function(sol), lista_unif, etiqueta_mod )
```

```
## [1] 12
```

```
sol <- ajusta_PLA(lista_unif,etiqueta_mod, 1000,rep(0,3))$hiperplane
count_errors(hiperplane_to_function(sol), lista_unif, etiqueta_mod )
```

```
## [1] 13
```

Conteo de errores función cuadrática

```
sol <- ajusta_PLA(lista_unif,etiqueta_mod_1, 10,rep(0,3))$hiperplane
count_errors(hiperplane_to_function(sol), lista_unif, etiqueta_mod )
```

```
## [1] 29
```

```
sol <- ajusta_PLA(lista_unif,etiqueta_mod_1, 100,rep(0,3))$hiperplane
count_errors(hiperplane_to_function(sol), lista_unif, etiqueta_mod )
```

```
## [1] 22
```

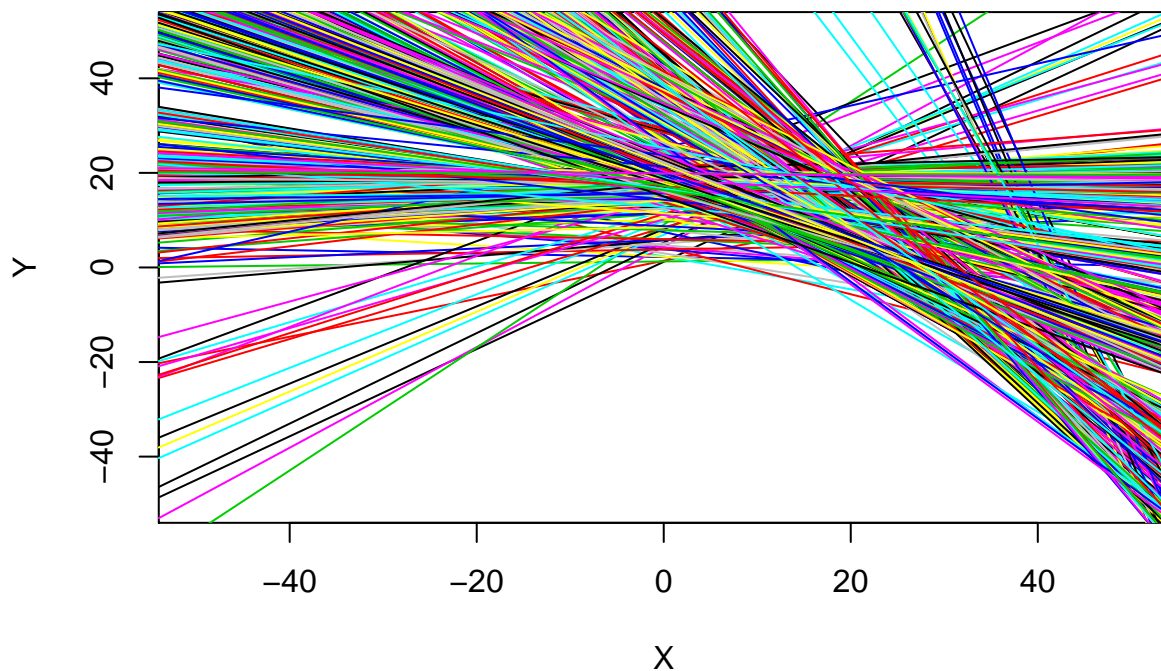
```
sol <- ajusta_PLA(lista_unif,etiqueta_mod_1, 1000,rep(0,3))$hiperplane
count_errors(hiperplane_to_function(sol), lista_unif, etiqueta_mod )
```

```
## [1] 39
```

Muestra de iteraciones

```
sol <- ajusta_PLA(lista_unif,etiqueta_mod, 10,rep(0,3), TRUE)$hiperplane
sol <- ajusta_PLA(lista_unif,etiqueta_mod, 100,rep(0,3), TRUE)$hiperplane
sol <- ajusta_PLA(lista_unif,etiqueta_mod, 1000,rep(0,3), TRUE)$hiperplane
```

Iteraciones PLA



Algoritmo PLA Modificado

```
ajusta_PLA_MOD <- function(datos, label, max_iter, vini, draw_iterations = FALSE){
  sol <- vini
  iter <- 0
  changed <- TRUE

  if(draw_iterations){
    plot.new()
    plot.window(xlim=c(-50,50), ylim=c(-50,50))
    axis(1)
```

```

axis(2)
box()
title(main="Iteraciones PLA", ylab = "Y", xlab = "X")
}

while ( iter < max_iter && changed){
  current_errors <- count_errors(hiperplane_to_function(sol), datos, label )
  current_sol <- sol
  changed <- FALSE
  for( inner_iter in 1:ncol(datos) ){
    x <- c(datos[,inner_iter],1)

    if( sign(crossprod(x,sol)) != label[inner_iter]){
      current_sol <- current_sol + label[inner_iter] * x
      changed <- TRUE
    }
  }

  if( current_errors >= count_errors(hiperplane_to_function(sol), datos, label )){
    sol <- current_sol
  }

  if(draw_iterations) abline(-sol[3]/sol[2], -sol[1]/sol[2], col=iter)
  iter <- iter+1
}
sol <- sol/sol[length(sol)-1]

return( list( hiperplane = sol, iterations = iter))
}

```

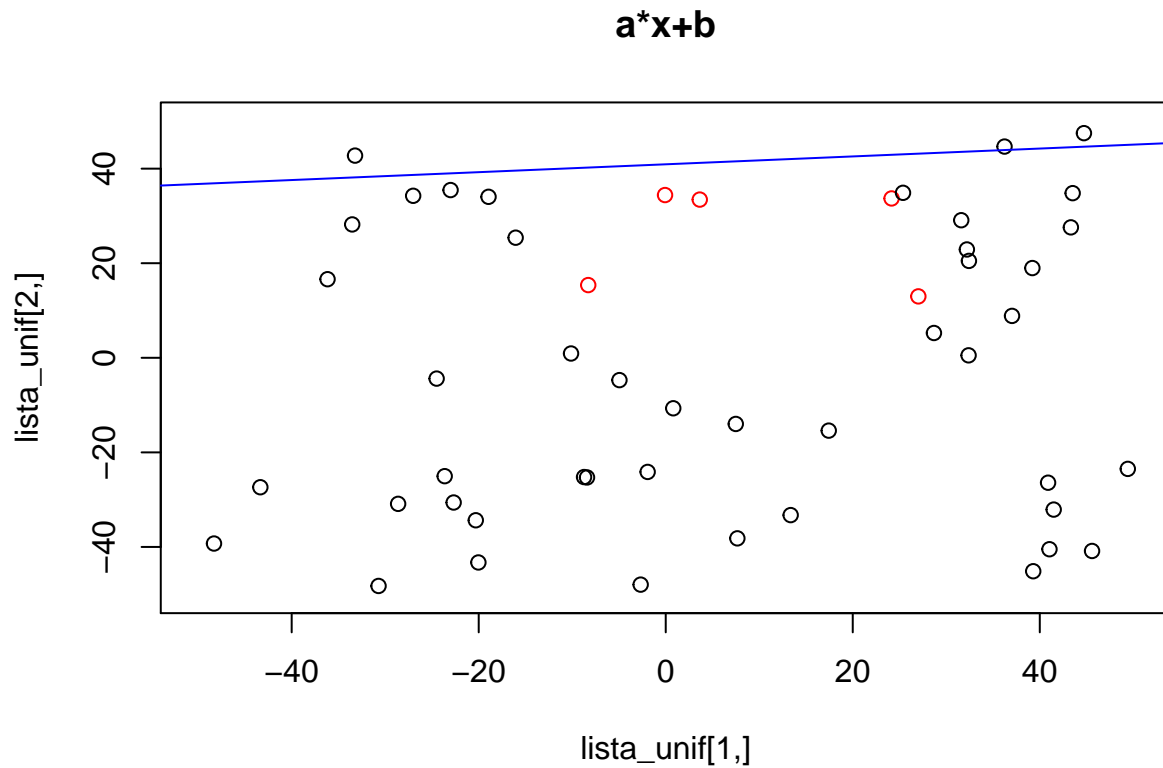
Evaluación funciones cuadráticas

```

plot.new()
plot.window(xlim=c(-50,50), ylim=c(-50,50))
axis(1)
axis(2)
box()
title(main="a*x+b", ylab = "lista_unif[2,]", xlab = "lista_unif[1,]")
points(lista_unif[1,etiqueta_1==-1], lista_unif[2,etiqueta_1==-1], col = 2)
points(lista_unif[1,etiqueta_1==1], lista_unif[2,etiqueta_1==1])

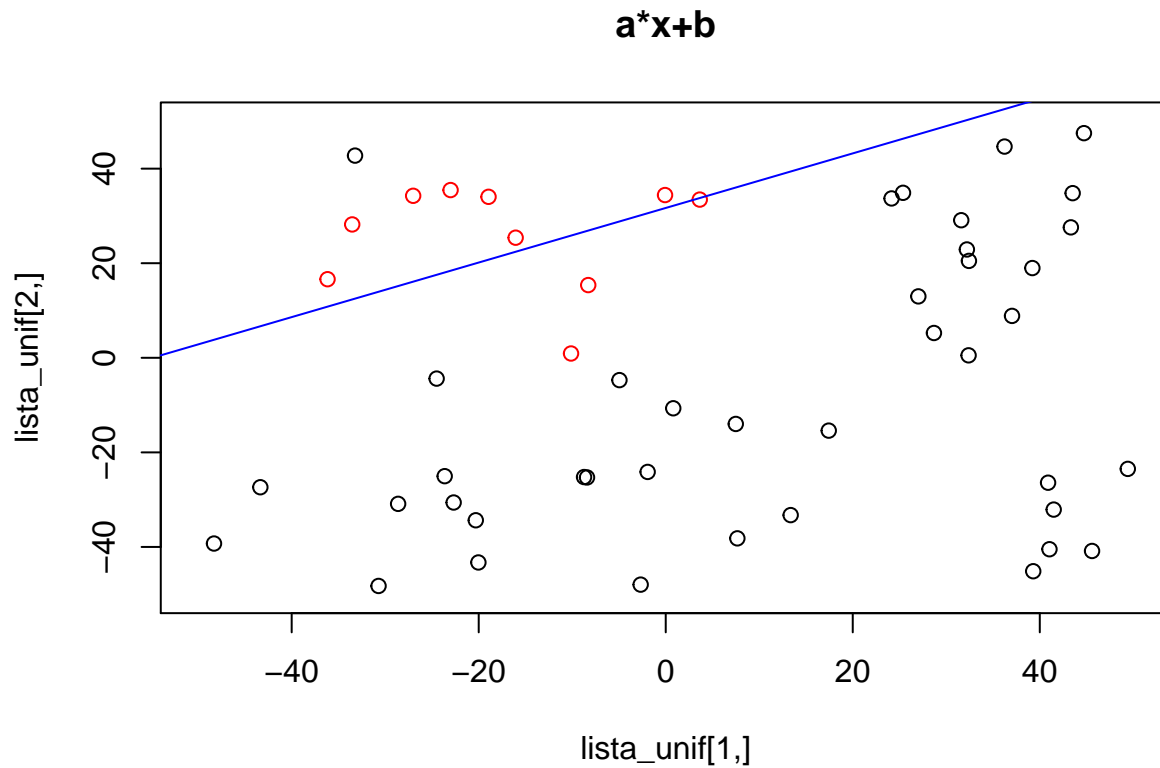
sol_cuadratic_1 <- ajusta_PLA_MOD(lista_unif, etiqueta_1, 1000, rep(0,3))$hiperplane
abline(-sol_cuadratic_1[3], -sol_cuadratic_1[1], col = 4)

```

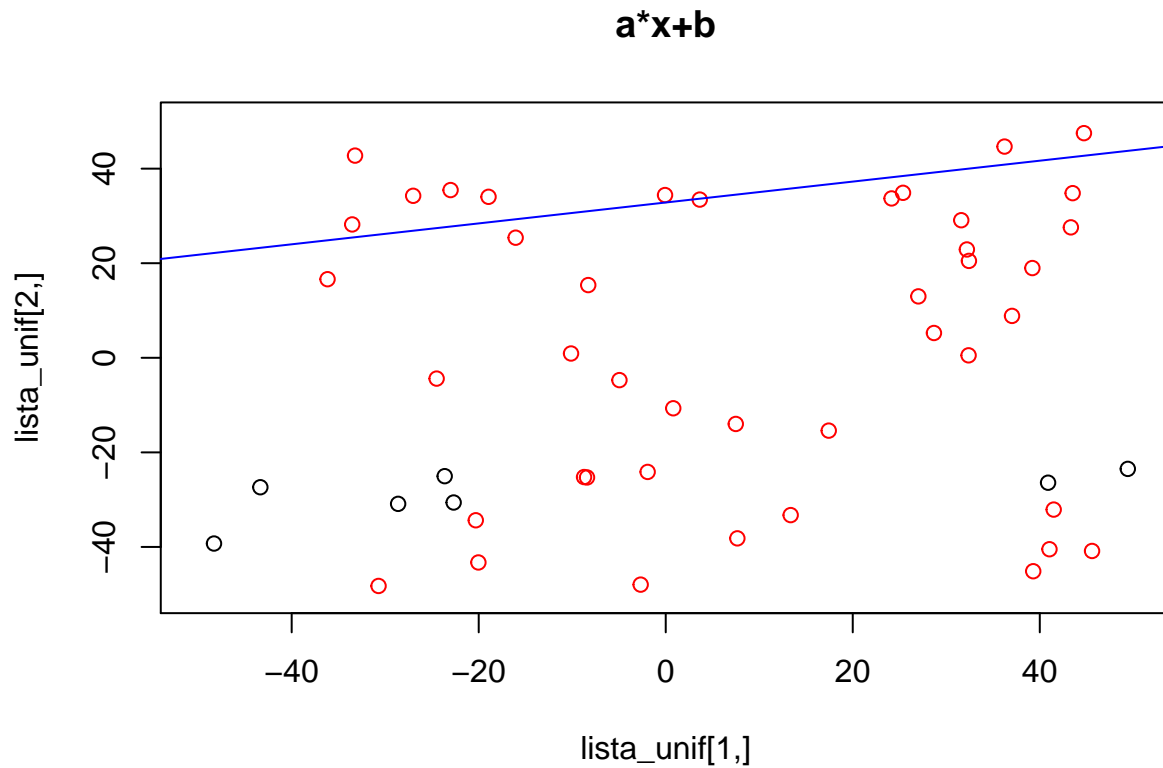
```
plot.new()
plot.window(xlim=c(-50,50), ylim=c(-50,50))
axis(1)
axis(2)
box()
title(main="a*x+b", ylab = "lista_unif[2,]", xlab = "lista_unif[1,]")
points(lista_unif[1,etiqueta_2==1], lista_unif[2,etiqueta_2==1], col = 2)
points(lista_unif[1,etiqueta_2==1], lista_unif[2,etiqueta_2==1])

sol_cuadratic_2 <- ajusta_PLA_MOD(lista_unif, etiqueta_2, 1000, rep(0,3))$hiperplane
abline(-sol_cuadratic_2[3], -sol_cuadratic_2[1], col = 4)
```



```
plot.new()
plot.window(xlim=c(-50,50), ylim=c(-50,50))
axis(1)
axis(2)
box()
title(main="a*x+b", ylab = "lista_unif[2,]", xlab = "lista_unif[1,]")
points(lista_unif[1,etiqueta_3==1], lista_unif[2,etiqueta_3==1], col = 2)
points(lista_unif[1,etiqueta_3==1], lista_unif[2,etiqueta_3==1])

sol_cuadratic_3 <- ajusta_PLA_MOD(lista_unif, etiqueta_3, 1000, rep(0,3))$hiperplane
abline(-sol_cuadratic_3[3], -sol_cuadratic_3[1], col = 4)
```



```
plot.new()
plot.window(xlim=c(-50,50), ylim=c(-50,50))
axis(1)
axis(2)
box()
title(main="a*x+b", ylab = "lista_unif[2,]", xlab = "lista_unif[1,]")
points(lista_unif[1,etiqueta_4==1], lista_unif[2,etiqueta_4==1], col = 2)
points(lista_unif[1,etiqueta_4==1], lista_unif[2,etiqueta_4==1])

sol_cuadratic_4 <- ajusta_PLA_MOD(lista_unif, etiqueta_4, 1000, rep(0,3))$hiperplane
abline(-sol_cuadratic_4[3], -sol_cuadratic_4[1], col = 4)
```

