Universidad Rey Juan Carlos

Grado en Matemáticas

Práctica 8

Geometría Computacional

Autores: Guillermo Grande Santi y
Alejandro López Adrados

Abril, 2022

Índice

1 Objetivos

El objetivo de esta práctica es la utilización de algún algoritmo, en este caso el algoritmo de Elzinga y Hearn, para la resolución de un problema 1-centro. Dicho problema 1-centro consiste en lo siguiente:
Si un conjunto S de n puntos del plano representa las ubicaciones de una serie de clientes o usuarios, el punto p del plano que minimiza la mayor de sus distancias a los elementos de S se llama el 1-centro euclídeo de S, y es la ubicación óptima, por ejemplo, para un servicio de naturaleza urgente, como puede ser un hospital. **Hallar este punto p es equivalente a encontrar el círculo de menor radio que cubre el conjunto S y tomar su centro p.**

2 Metodología

Para resolver este problema hemos decidido usar el algoritmo de Elzinga y Hearn. En primer lugar, crearemos una serie de numeros aleatorios (este número puede cambiarse con la variable elementos) para usar el algoritmo con distintos conjuntos de puntos (llamemos S al conjunto de puntos de cada ocasión).
El primer paso de este algoritmo es escoger una pareja de puntos. que llamaremos punto1 y punto2, y construir el círculo C de diámetro punto1-punto2. En este momento, se recorren con un bucle todos los puntos para ver si están dentro del círculo. Si todos los puntos están en su interior, entonces hemos encontrado el círculo mínimo. En caso contrario, se pasará al paso 4 tomando un punto3 de fuera del círculo.

En este paso se creará el triángulo formado por punto1-punto2-punto3, se calcularán sus ángulos interiores y se verá qué clase de triángulo se ha formado. Si el triángulo es obtuso o rectángulo ($\alpha \geq \pi/2$), los puntos más alejados del ángulo obtuso o rectángulo se renombrarán a punto1 y punto2 y se volverá al paso 2. Si el triángulo es agudo, se construirá un nuevo círculo C que pasará por los tres puntos. Esto último se hará mediante la resolución de un sistema

basado en la ecuación de la circunferencia en su forma general, que es:

$$x^2 + y^2 + Dx + Ey + F = 0$$

Posteriormente, se pasará al paso 5.

El paso 5 es una repetición del paso 3 pero con la nueva circunferencia, es decir, se comprobará si todos los puntos están en el interior de esta circunferencia. En caso afirmativo, tenemos el círculo mínimo buscado, y en caso contrario, se tomará otro punto punto4 fuera de la circunferencia y se pasará al paso 6.

Ahora se tomará el punto de los tres primeros (punto1-punto2-punto3) que esté más alejando del punto4. Para ello se usará la funcion distancia(), que se encarga de calcular la norma de cada vector que une un punto de los tres primeros con punto4. Este punto será Q, y uniéndose con el centro de la circunferencia obtendremos una recta que nos servirá en el siguiente paso.

A continuación, nos basaremos para el paso 7 en la utilización de la función encimaDebajo() (Esta función es una simplificación de la práctica 7.2), ya que debemos buscar el punto del conjunto punto1-punto2-punto3 que esté en el lado opuesto de punto4 en relación a la recta obtenida en el paso anterior. Por ello, a la función se le pasará como parámetros Q, el centro de la circunferencia y punto4 (que en este punto del programa se ha renombrado a puntofuera). Una vez realizadas las comprobaciones necesarias obtendremos el punto 8 (opuesto a puntofuera), al cual llamaremos R y volveremos a llamar al paso 4 con los nuevos parámetros que describirán una nueva circunferencia a probar (punto1 = Q, punto2 = R, punto3 = puntofuera). Esto se repetirá sucesivamente hasta que hayamos encontrado el círculo mínimo.

En ese instante, se procederá a pintar el círculo encontrado y el centro de dicho círculo. Además, los atributos del círculo mínimo se pintarán por pantalla (centro y radio), conociendo así el centro, que es la solución al problema 1-centro.

3 Conclusiones

En la resolución de este ejercicio hemos aprendido la utilización de otro algoritmo, el cual es muy vistoso en la posterior representación. Para llegar al resultado final nos hemos ido encontrando con diferentes obstáculos que hemos tenido que ir refrescando como la creación de un círculo a partir de tres puntos o cómo evaluar si un punto se encuentra en el lado opuesto respecto de una recta de otro punto.

Sin embargo, como valoración general ha sido un algoritmo sencillo de implementar en comparación con el de la anterior práctica (Algoritmo Scan de Graham) lo que indica una buena progresión en este lenguaje y en la asignatura en su conjunto.

4 Anexos

4.1 Código del ejercicio

```
#Algoritmo de Elzinga y Hearn 1-centro


encimaDebajo <- function(A, B, C)
{
  r <- function(x) {
    y = (((B[2] - A[2]) / (B[1] - A[1])) * (x - A[1])) + A[2]
  }


  x <- C[1]
  y <- r(C)
  pendiente <- (B[2] - A[2]) / (B[1] - A[1])
  if (A[2] == B[2]) {
    s <- 'Recta horizontal, ni a derecha ni a izquierda'
  }else{
```

```
    if (((y[1] < C[2]) & (pendiente > 0)) |
        (y[1] > C[2]) & (pendiente < 0))
      s <- 1
    else if (((y[1] > C[2]) & (pendiente > 0)) |
             (y[1] < C[2]) & (pendiente < 0))
      s <- 2
    if (y[1] == C[2])
      s <- 3
  }

  s
}

distancia <- function(punto1, punto2) {
  sqrt((punto2[1] - punto1[1]) ^ 2 + (punto2[2] - punto1[2]) ^ 2)
}

paso2 <- function(puntos,punto1,punto2) {
  dimension <- dim(puntos)
  centro <-
    c((punto1[1] + punto2[1]) / 2, (punto1[2] + punto2[2]) / 2)
  diametro <- distancia(punto1, punto2)
  radio <- diametro / 2
  circulo <- function(x) {
    y = sqrt(radio ^ 2 - (x - centro[1]) ^ 2) + centro[2]
  }
  x <- seq(min(punto1[1], punto2[1]), max(punto1[1], punto2[1]))

  #PASO 3
  h <- 1
```

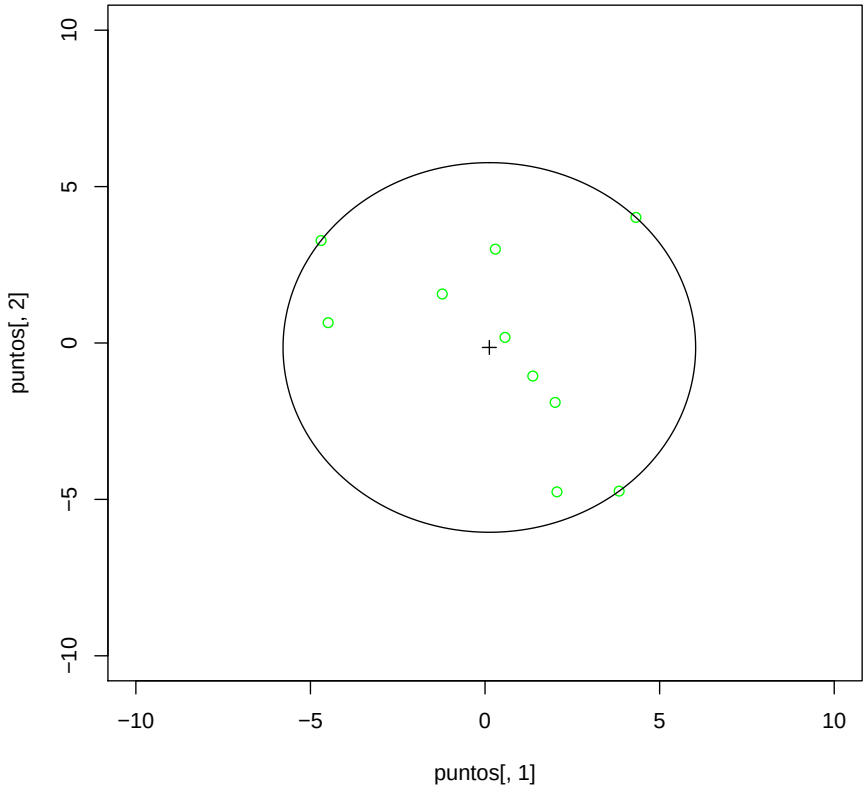
```
  cont <- 0
  for (h in 1:dimension[1]) {
      punto3 <- puntos[h, ]
      x<-(punto3[1] - centro[1]) ^ 2
      y<-(punto3[2] - centro[2]) ^ 2
      r2<-radio^2+0.001
      izq<-x+y
      if ((izq<r2) | (izq==r2)){
        cont <- cont + 1
      } else {
        puntofuera <- punto3
      }
  }
  if (cont == dimension[1]) {
    circMinimo <- c(centro[1], centro[2], radio)
    t<-seq(0,2*pi,0.001)
    x<-sin(t)*radio+centro[1]
    y<-cos(t)*radio+centro[2]
    lines(x,y)
    circMinimo
  } else {
    paso4(punto1, punto2, puntofuera,puntos)
  }
}

paso4 <- function(punto1, punto2, punto3,puntos) {
  #Hcaemos un triángulo con punto1,punto2,punto3
  lado1 <- c(punto2[1] - punto1[1], punto2[2] - punto1[2])
  lado2 <- c(punto3[1] - punto1[1], punto3[2] - punto1[2])
  lado3 <- c(punto3[1] - punto2[1], punto3[2] - punto2[2])
```

```
angulo12 <- acos((lado1 %*% lado2) /
                   (sqrt(lado1[1] ^ 2 + lado1[2] ^ 2) *
                    sqrt(lado2[1] ^ 2 + lado2[2] ^ 2)))
angulo13 <- acos((-lado1 %*% lado3) /
                   (sqrt(lado1[1] ^ 2 + lado1[2] ^ 2) *
                   sqrt(lado3[1] ^ 2 + lado3[2] ^ 2)))
angulo23 <- acos((-lado2 %*% -lado3) /
                   (sqrt(lado2[1] ^ 2 + lado2[2] ^ 2) *
                   sqrt(lado3[1] ^ 2 + lado3[2] ^ 2)))
if (angulo12 >= pi / 2) {
  paso2(puntos,punto2, punto3)
} else if (angulo13 >= pi / 2) {
  paso2(puntos,punto1, punto3)
} else if (angulo23 >= pi / 2) {
  paso2(puntos,punto1, punto2)
} else {
  #Hacemos el circulo con punto1, punto2, punto3
  A <- matrix(nrow = 3, ncol = 3)
  A[1,] <- c(punto1[1], punto1[2], 1)
  A[2,] <- c(punto2[1], punto2[2], 1)
  A[3,] <- c(punto3[1], punto3[2], 1)
  b <- matrix(nrow = 3, ncol = 1)
  b[1] <- -(punto1[1] ^ 2 + punto1[2] ^ 2)
  b[2] <- -(punto2[1] ^ 2 + punto2[2] ^ 2)
  b[3] <- -(punto3[1] ^ 2 + punto3[2] ^ 2)
  x <- solve(A, b)
  centro <- c()
  centro[1] <- x[1] / -2
  centro[2] <- x[2] / -2
  radio <- sqrt(centro[1] ^ 2 + centro[2] ^ 2 - x[3])
```

```
#PASO 5
dimension<-dim(puntos)
h <- 1
cont2 <- 0
for (h in 1:dimension[1]) {
    punto4 <- puntos[h, ]
    x<-(punto4[1] - centro[1]) ^ 2
    y<-(punto4[2] - centro[2]) ^ 2
    r2<-radio^2+0.001
    izq<-x+y
    if ((izq<r2) | (izq==r2)){
      cont2 <- cont2 + 1
    } else {
      puntofuera <- punto4
    }
  }
if (cont2 == dimension[1]) {
  circMinimo <- c(centro[1], centro[2], radio)
  t<-seq(0,2*pi,0.001)
  x<-sin(t)*radio+centro[1]
  y<-cos(t)*radio+centro[2]
  lines(x,y)
  circMinimo
} else{
  #PASO 6
  distancia1 <- distancia(punto1, puntofuera)
  distancia2 <- distancia(punto2, puntofuera)
  distancia3 <- distancia(punto3, puntofuera)
  if ((distancia1 >= distancia2) & (distancia1 >= distancia3)) {
    q <- punto1
```

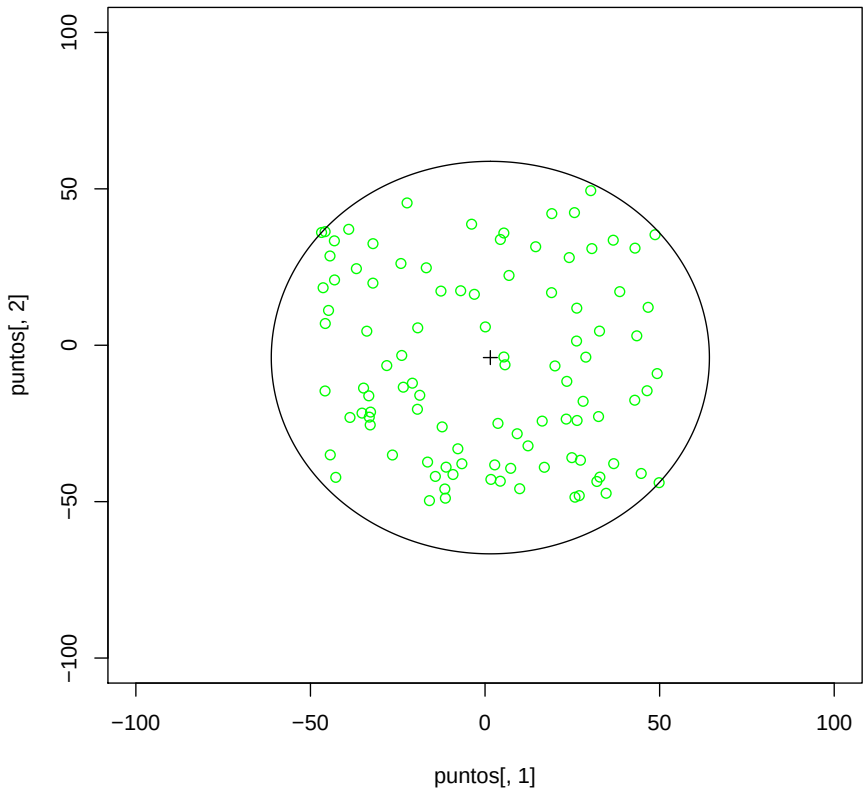
```
      } else if ((distancia2 >= distancia1) &
                 (distancia2 >= distancia3)) {
        q <- punto2
      } else{
        q <- punto3
      }
      #PASO 7
      if (encimaDebajo(q, centro, puntofuera) == 1) {
        if (encimaDebajo(q, centro, punto1) == 2) {
          r <- punto1
        } else if (encimaDebajo(q, centro, punto2) == 2) {
          r <- punto2
        } else if (encimaDebajo(q, centro, punto3) == 2) {
          r <- punto3
        }
      } else if (encimaDebajo(q, centro, puntofuera) == 2) {
        if (encimaDebajo(q, centro, punto1) == 1) {
          r <- punto1
        } else if (encimaDebajo(q, centro, punto2) == 1) {
          r <- punto2
        } else if (encimaDebajo(q, centro, punto3) == 1) {
          r <- punto3
        }
      }
      #PASO 8
      paso4(q, r, puntofuera,puntos)
    }
  }
}
```

```
elzingaHearn <- function(puntos) {
  plot(
  puntos[, 1],
  puntos[, 2],
  xlim = c(-limite*2, limite*2),
  ylim = c(-limite*2, limite*2),
  type = "p",
  col = "green"
  )
  vResultado<-paso2(puntos,puntos[1,],puntos[2,])
  resultado<-rbind(vResultado)
  colnames(resultado)<-c("Coordenada x centro","Coordenada y centro", "Radio")
  points(resultado[1,1],resultado[1,2], cex=1, pch=3)
  resultado
}
#Ejemplo con 10 puntos de -5 a 5
elementos<-10
limite <- 5
vec1<-runif(elementos,-limite,limite)
vec2<-runif(elementos,-limite,limite)
vec<-matrix(nrow=elementos,ncol=2)
vec<-cbind(vec1,vec2)
puntos<-vec

elzingaHearn(puntos)
```



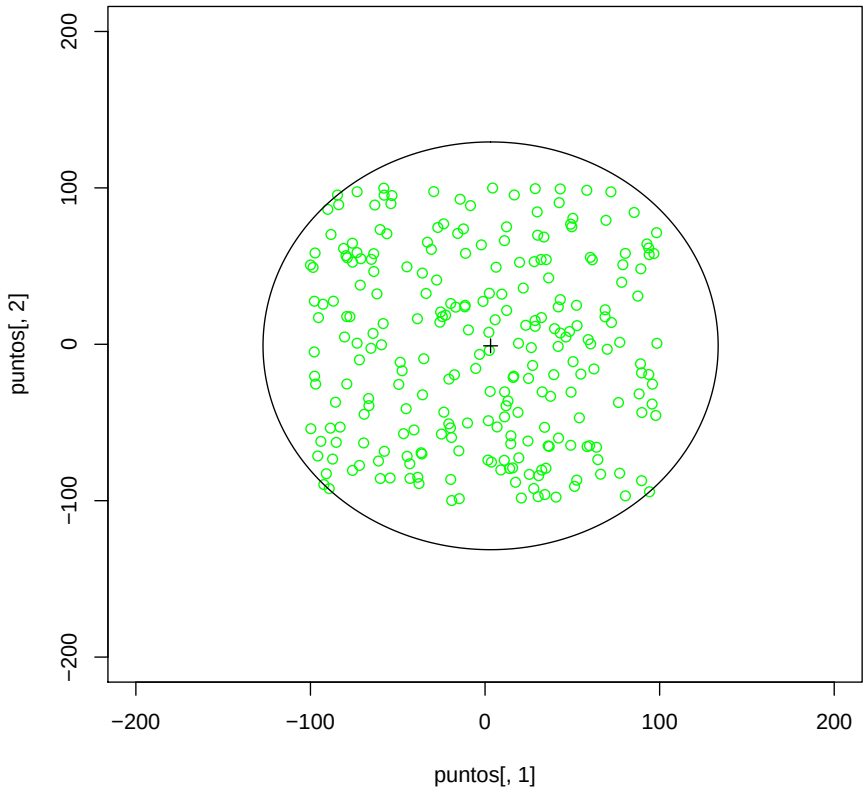

```
##            Coordenada x centro Coordenada y centro   Radio
## vResultado           0.1231831          -0.1426103 5.90803

#Ejemplo con 100 puntos de -50 a 50
elementos<-100
limite <- 50
vec1<-runif(elementos,-limite,limite)
vec2<-runif(elementos,-limite,limite)
vec<-matrix(nrow=elementos,ncol=2)
vec<-cbind(vec1,vec2)
puntos<-vec
```

```
elzingaHearn(puntos)
```




```
##            Coordenada x centro Coordenada y centro    Radio
## vResultado            1.517457           -3.943696 62.73692

#Ejemplo con 250 puntos de -100 a 100
elementos<-250
limite <- 100
vec1<-runif(elementos,-limite,limite)
vec2<-runif(elementos,-limite,limite)
vec<-matrix(nrow=elementos,ncol=2)
```

```r
vec<-cbind(vec1,vec2)
puntos<-vec


elzingaHearn(puntos)
```




```
##            Coordenada x centro Coordenada y centro    Radio
## vResultado            3.176094          -0.9834916 130.3447
```