Clustering IRIS

HILL CLIMBING

Clustering IRIS

Debemos dividir las 150 observaciones en 3 grupos.

Vamos a minimizar la distancia de cada observación al centroide de su grupo.

Se empleará la distancia Manhatan.

Toda observación debe pertenecer a un conjunto, y todo conjunto debe tener al menos una observación.

Se empleará hill climbing.

Clustering IRIS

Paso 0.

Lectura de datos. Guarden las medidas de cada observación en una matriz $Obs[150][4]$

Ahora normalicen los datos mediante la siguiente fórmula

$$Obs[i][j] = \frac{Obs[i][j] - C_{min}(j)}{C_{max}(j) - C_{min}(j)}$$

Donde

$Obs[i][j]$, es el valor de la observación *i* en la columna *j*

$C_{min}(j)$, es el mínimo valor de la columna *j*

$C_{max}(j)$, es el máximo valor de la columna *j*

obs1	5.1	3.5	1.4	0.2
obs2	4.9	3	1.7	0.4
obs3	7	3.2	4.7	1.4
obs4	6.4	3.2	4.5	1.5
obs5	5.8	2.7	5.1	1.9
obs6	7.1	3	5.9	2.1

Mínimo	4.9	2.7	1.4	0.2
Máximo	7.1	3.5	5.9	2.1

Normalizados

obs1	0.09090909	1	0	0
obs2	0	0.375	0.06666667	0.10526316
obs3	0.95454545	0.625	0.73333333	0.63157895
obs4	0.68181818	0.625	0.68888889	0.68421053
obs5	0.40909091	0	0.82222222	0.89473684
obs6	1	0.375	1	1

Clustering IRIS

Paso 1.

Representación de las soluciones:

$$Sol\_act[150] = (x_0, x_1, x_2, x_3, \dots, x_{148}, x_{149})$$

Donde $x_i \in \{0,1,2\}$.

Por ejemplo

$$Sol\_act[150] = (2,2,0,1, \dots, 1,1\ )$$

obs1	0.09090909	1	0	0	2
obs2	0	0.375	0.06666667	0.10526316	2
obs3	0.95454545	0.625	0.73333333	0.63157895	0
obs4	0.68181818	0.625	0.68888889	0.68421053	1
obs5	0.40909091	0	0.82222222	0.89473684	0
obs6	1	0.375	1	1	1

Clustering IRIS

Paso 2.

Generación de solución inicial.

$Sol\_Inicial()$

- $int\ Solucion[150]$
- For i = 0 to 149
- $\quad Solucion[i] \leftarrow rnd(0,1,2)$
- return($Solucion$)

Al final debe revisar que cada uno de estos valores aparece al menos una vez.

Clustering IRIS

Paso 3.

Generar la función objetivo.

$Func\_Obj(Solucion[150])$

- Calculamos los centroides de cada grupo
- $float\ C[3][4] = \bar{0}$ //Son 3 centroides, cada uno de 4 dimensiones
- $int\ contador[3] = \bar{0}$
- For i = 0 to 2
- For j = 0 to 149
- if $Solucion[j] == i$
- $C[i] \mathrel{+}= Obs[j]$
- $contador[i] \mathrel{+}= 1$
- For i = 0 to 2
- $C[i] = C[i]/contador[i]$ //Ya calculamos los centroides

Clustering IRIS

Paso 3. (continuación…)

- $float\ distancia = 0$
- For i = 0 to 2
- For j = 1 to 150
- if $Solucion[j] == i$
- $distancia \mathrel{+}= \sum_{k=0}^{3} abs(C[i][k] - Obs[j][k])$
- $return\ (distancia)$

Clustering IRIS

obs1	0.09	1	0	0	2
obs2	0	0.38	0.07	0.11	2
obs3	0.95	0.63	0.73	0.63	0
obs4	0.68	0.63	0.69	0.68	1
obs5	0.41	0	0.82	0.89	0
obs6	1	0.38	1	1	1

$$C[0] = [(0.95,0.63,0.73,0.63) + (0.41,0,0.82,0.89)]/2$$

$$= (0.68, 0.315, 0.775, 0.76)$$

Clustering IRIS

obs1	0.09	1	0	0	2
obs2	0	0.38	0.07	0.11	2
obs3	0.95	0.63	0.73	0.63	0
obs4	0.68	0.63	0.69	0.68	1
obs5	0.41	0	0.82	0.89	0
obs6	1	0.38	1	1	1

$$C[1] = [(0.68\ , 0.63, 0.69, 0.68) + (1, 0.38, 1, 1)]/2$$

$$= (0.84, 0.505, 0.845, 0.84)$$

Clustering IRIS

obs1	0.09	1	0	0	2
obs2	0	0.38	0.07	0.11	2
obs3	0.95	0.63	0.73	0.63	0
obs4	0.68	0.63	0.69	0.68	1
obs5	0.41	0	0.82	0.89	0
obs6	1	0.38	1	1	1

$C[0] = (0.68, 0.315, 0.775, 0.76)$

$$distancia \mathrel{+}= abs(0.68 - 0.95) + abs(0.315 - 0.63)$$
$$+abs(0.775 - 0.73) + abs(0.76 - 0.63)$$

Clustering IRIS

obs1	0.09	1	0	0	2
obs2	0	0.38	0.07	0.11	2
obs3	0.95	0.63	0.73	0.63	0
obs4	0.68	0.63	0.69	0.68	1
obs5	0.41	0	0.82	0.89	0
obs6	1	0.38	1	1	1

$C[1] = (0.84, 0.505, 0.845, 0.84)$

$$distancia \mathrel{+}= abs(0.84 - 0.68) + abs(0.505 - 0.63)$$
$$+abs(0.845 - 0.69) + abs(0.84 - 0.68)$$

Clustering IRIS

Paso 4.

Main()

- $float\ Obs[150][4]$
- Leer, cargar y normalizar los datos
- $Sol\_act[150]$, $Vecinos[3][150]$
- $float\ Costo\_act$
- $Sol\_act = Sol\_Inicial()$ //Generar solución inicial
- $Costo\_act = Func\_Obj(Sol\_act)$
- ……

- // Inicia hill climbing
- $For\ it = 0\ to\ 100$
- //Generamos y evaluamos todos los vecinos
- $for\ j = 0\ to\ 150$

- $for\ i = 0\ to\ 3$
- $if\ i\ ! = Sol\_act[j]$
- $aux = Sol\_act[j]$
- $Sol\_act[j] = i$
- $Vecinos[i][j] = Func\_Obj(Sol\_act)$
- $Sol\_act[j] = aux$
- $else$
- $Vecinos[i][j] = 1000000$
- Sea $Vecinos[m][n]$ una solución vecina de menor costo
- $if\ Vecinos[m][n] \leq Costo\_act$
- $Sol\_act[n] = m$
- $Costo\_act = Vecinos[m][n]$
- $print(Costo\_act)$
- $print(Soll\_act)$
-