Whale Optimziation Algorithm aplicado al problema de clustering

Yábir García Benchakhtir

24 de junio de 2020

Índice

1. Introducción al problema

Consideramos el problema del agrupamiento con restricciones. En este problema contamos con un conjunto no vacio $P \subset \mathbb{R}^n$ de puntos y nos planteamos cómo podríamos agruparlos de manera que exista una relación entre los puntos de un mismo grupo. A cada grupo lo denominaremos *cluster* y notaremos $\mathcal{C}$ al conjunto de todos los clusters.

Sobre esta base imponenmos restricciones en la manera en la que se realizan las agrupaciones. En primer lugar existe un subconjunto de pares de puntos ML definido como

$$ML = \{(a, b) \in P \times P \mid \quad a \in K \iff b \in K \text{ para } K \in \mathcal{C}\}$$

es decir, el conjunto de puntos que han de estar en el mismo cluster. De manera similar existe otro conjunto CL de pares de puntos que no pueden pertenecer al mismo conjunto.

$$CL = \{(a, b) \in P \times P \mid \quad a \in K \iff b \notin K \text{ para } K \in \mathcal{C}\}$$

Notaremos por $R = ML \cup CL$ al conjunto de restricciones del problema.

Nos concentraremos en encontrar soluciones bajo restricciones *débiles* a este problema donde intentaremos encontrar soluciones que minimicen el conjunto restricciones violadas.

En nuestros algoritmos intentaremos minimizar la distancia que haya entre los puntos de cada cluster sujeto a que se violen el menor número de restricciones. Vamos a formalizar pues esta idea, definimos el centroide de un cluster como el punto promedio de los puntos del cluster

$$\mu_i = \frac{1}{|c_i|} \sum_{x \in c_i} x \quad \text{con } c_i \in \mathcal{C} \text{ para todo } i \in \{1, ...k\}$$

Definimos la distancia *intra-cluster* como la media de las distancias de cada punto del cluster al centroide. En este caso consideramos la distancia euclidea.

$$\bar{c}_i = \frac{1}{|c_i|} \sum_{x \in c_i} ||x - \mu_i||_2 \quad \text{con } c_i \in \mathcal{C} \text{ para todo } i \in \{1, ...k\}$$

La desviación general del problema será por tanto

$$\bar{C} = \frac{1}{k} \sum_{c_i \in \mathcal{C}} \bar{c}_i$$

Nuestro objetivo cuanto busquemos una solución con restricciones débiles será minimizar la desviación general sujeto a que "no incumplamos demasiadas restricciones". Para obtener un valor que le de mayor o menor relevancia a la cantidad de restricciones que incumplamos usaremos la siguiente proporción

$$\lambda = \frac{\lceil d \rceil}{|R|}$$

donde d es la mayor distancia entre puntos del conjunto de puntos P. Así nuestro objetivo,cuando busquemos una solución con restricciones débiles, será minimizar

$$f = \bar{C} + \lambda * \text{infeasibility}$$

donde *infeasibility* es la cantidad de restricciones incumplidas.

2. Whale Optimziation Algorithm

La metaheurística elegida para resolver el problema ha sido Whale Optimziation Algorithm (WOA) propuesta por Seyedali Mirjalili y Andrew Lewis en 2016. Esta metaheurística basa su comportamiento en las técnicas depredadoras de la ballena jorobada y su comportamiento social.

Utilizando el especial comportamiento que tiene este animal cuando colabora con otros de su misma especia se pretende conseguir una metaheurística que proporcione buenos resultados en problemas de optimización de funciones reales intentando preservar un equilibrio entre exploración y explotación.

Más concretamente la técnica de caza consiste en crear una espiral entorno a la presa y levantar un *muro* de burbujas de aire, haciendo que esta se desoriente para posteriormente acercarse y atacar.

2.1. Descripción de la metaheurística

Para modelar el problema los autores de la metaheurística proponen un modelado matemático del comportamiento de la ballena jorobada que se pueda adaptar adaptar a la optimización de una función real.

Los agentes X que participan en nuestro algoritmo (y que representan a las ballenas) se van a representar como N vectores de dimensión d donde cada componente del vector del agente representa una coordenada en nuestra solución

$$X_i(t) =< X_{i1}(t), X_{i2}(t), \ldots, X_{id}(t) >$$

como deja entrever esta notación el estado de la ballena depende de una variable temporal t que represneta el instante de tiempo en el que nos encontramos y que está limitado por una constante T que fijamos nosotros. En cada instante t la mejor solución vendrá represnetada por X^*.

Procedemos en primer lugar a definir el operador distancia entre dos ballenas

$$\begin{aligned} &||\cdot||\colon \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n \\ &\qquad (< x_1, x_2, \ldots, x_n >, < y_1, y_2, \ldots, y_n >) \mapsto < |x_1 - y_1|, |x_2 - y_2|, \ldots, |x_n - y_n| > \end{aligned}$$

que es la distancia coordenada a coordenada.

El movimiento de caza se represneta por la modificación del vector agente mediante la expresión

$$X_i(t+1) = X^*(t) - A \cdot D_i^1 \tag{1}$$

donde $\cdot$ representa el producto componente a componente de $\mathbb{R}^n$, D viene dado por la expresión

$$D_i^1 = ||CX^*(t) - X_i(t)||$$

y A y C se obtinen como

$$\begin{aligned} A &= 2a \cdot r - a \\ C &= 2r \end{aligned}$$

con $r \in [0,1]^d$ un vector aleatorio y $a \in [0,2]^d$ constante que se hace decrecer de manera lineal a lo largo de los distintos pasos del algoritmo mediante la ecuación

$$a(t) = 2 - 2\frac{t}{\text{max\_evaluaciones}}$$

La técnica de *caza* se basa en combinar este movimiento que nos proporciona un componente de exploración junto a la creación de una espiral mediante la actualización del agente de acuerdo a la expresión

$$\begin{cases} X_i(t+1) & = e^{bl} cos(2\pi l) D_2 + X^* \\ D_i^2 & = ||X^*(t) - X_i(t)|| \end{cases} \tag{2}$$

siendo $b \in \mathbb{R}$ constante y $l \in [-1, 1]$ aleatorio de manera que nos definen un radio para la espiral en cada instante. La notación $\oplus$ se define como

El movimiento de caza natural mezcla tanto los desplazamientos en linea recta como el comportamiento en espiral por lo que se introduce un factor aleatorio $p \in [0, 1]$ que decide que tipo de movimeinto se va a realizar

$$X_i(t+1) = \begin{cases} X^*(t) - A \cdot D_i^1 & p < \frac{1}{2} \\ e^{bl} cos(2\pi l) D_i^2 + X^* & p \geq \frac{1}{2} \end{cases}$$

2.2. Adaptación de la metaheurística al problema

Durante el desarrollo de la asignatura hemos trabajado con una representación de la solución que se centraba en la asignación de cada punto a un cluster y se recalculaba en cada caso los centros de cada cluster. El espacio de búsqueda era

$$\{< x_1, x_2, \dots x_n >: x_i \in [0, k] \cap \mathbb{N}, k > 0\}$$

donde n representa el número de puntos que intervienen en el problema y k el número de clusters que consideramos.

Para adaptar la metaheurística he decidido variar mi enfoque del problema y, en lugar de modificar las asignaciones que hago de los puntos, pensar que cada agente representa las coordenas de los centroiedes del problema.

Así cada agente (ballena) queda definido como

$$X_i = << c_{01}, c_{02}, \dots c_{0d} >, \overset{(k)}{\dots}, < c_{k1}, c_{k2}, \dots c_{kd} >>$$

una lista de k vectores con dimensión d, la dimensión de los puntos del problema. No obstante esto no es suficiente ya que nuestro objetivo final es proporcionar una asignación de los puntos a cada cluster.

El algoritmo WOA se encargará de minimizar la componente real y continua de la métrica que estamos evaluando (que además tiene más peso en la misma), la distancia de los centroides respecto a los puntos de cada cluster. La segunda componente de la métrica es el factor de *infeasibility* que minimizaremos asignando a cada punto el cluster que menos incremento en infeasibility produzca y, en caso de empate, el cluster más cercano.

El comportamiento del algoritmo, como se muestra a continuación, es sencillo

Algorithm 1 Whale optimization Algorithm

```
1: procedure WOA(max_evaluaciones)
2:     whales: Inicializar un conjunto de ballenas con k centroides aleatorios
3:     Evaluamos las diferentes ballenas usando nuestra métrica
4:     Selecccionamos X* la mejor ballena
5:     Guardamos la mejor solución encontrada al problema
6:     evaluaciones ← 0
7:     while evaluaciones < max_evaluaciones do
8:         actualizamos el parametro a
9:         for agente en la lista de ballenas do
10:            Determianr p y calcular A y C.
11:            if p < 0,5 then
12:                if |A| < 1 then
13:                    Movemos la ballena usando el movimeinto rectilineo (1)
14:                else if |A| > 1 then
15:                    Movimiento rectilineo usando una ballena eleatorio
16:                end if
17:            else
18:                Movimiento en espiral utilizando (2)
19:            end if
20:        end for
21:        Comprobar si alguna solución se ha salido de los limites del problema
22:        Incremenetar evaluaciones en el número de evaluaciones correspondiente
23:        Evaluar las soluciones encontradas y actualizar la mejor ballena
24:        if la mejor ballena es la mejor solución encontrada then
25:            Actualizar la mejor solución encontrada hasta el momento
26:        end if
27:    end while
28:    return Construir la solución asociada a la mejor ballena encontrada
29: end procedure
```

El algoritmo que, a partir de una ballena, nos permite crear una solución en el formato que hemos descrito es

Algorithm 2 Generar una solución a partir de un conjunto de centroides

```
1: procedure GENERAR SOLUCIÓN(datos, restricciones, centroides)
2:     clusters ← [[] . . . []] Lista con los puntos asociados a cada cluster
3:     for cada punto p del conjunto de datos do
4:         Calcular el coste de infeasibility que supone asignar el punto a los diferentes clusters
5:         elegir K = [k_0, . . . , k_s] los clusters que menor incremento de infeasibility suponen
6:         if K.size() == 1 then
7:             Asignar p al cluster K[0]
8:         else
9:             Calcular la dinstancia del punto p al centro de cada cluster k_i ∈ K
10:            Asignar p al cluster más cercano
11:        end if
12:    end for
13:    return Construir la solución asociada a la mejor ballena encontrada
14: end procedure
```

3. Implementación y desarrollo del experimento

La implementación del código ha sido realizada en el lenguaje de programación *Rust* ya que ha sido el el usado durante el desarrollo de las anteriores prácticas, por lo que se ha podido reutilzar código y las ejecuciones han sido realizadas bajo las mismas optimzaciones en el lenguaje.

Los experimentos han sido ejecutados en un ordenador con las siguientes características

- CPU: Ryzen 5 2600 3.4GHz
- RAM: 24GB
- SO: Solus OS

Cada experimento consistente en la ejecución del algoritmo bajo 5 semillas distintas (37,42,440,699,752) de las que se han recopilado datos y se han agreado tomando medias. También se proporcionan en detalle los resultados para cada ejecución.

El tamaño de la población como hemos estudiado para este tipo de problemas se recomienda entre 20 y 50 individuos. En mi caso he optado por fijar este valor a 30 individuos a fin de obtener una primera idea del funcionamiento del algoritmo. El parametro b de la espiral lo he fijado a 1 ya que es el valor que se fija en la versión implementada en MATLAB. Tras haber ejecutado el algoritmo los resultados son los siguientes.

	Iris	Ecoli	Rand	Tyroid
Semilla 37	0.670304	13.046238	0.715599	14.013687
Semilla 42	0.671846	13.140929	0.715599	13.500604
Semilla 440	0.656589	13.058654	0.715599	13.295182
Semilla 699	0.668911	13.050045	0.715599	13.647765
Semilla 752	0.666602	13.026891	0.715599	13.790620
Media	0.666851	13.064551	0.715599	13.649572

(a) Distancia intracluster

	Iris	Ecoli	Rand	Tyroid
Semilla 37	42.000000	1273.000000	0.000000	185.000000
Semilla 42	32.000000	1285.000000	0.000000	204.000000
Semilla 440	39.000000	1298.000000	0.000000	203.000000
Semilla 699	48.000000	1288.000000	0.000000	206.000000
Semilla 752	40.000000	1304.000000	0.000000	191.000000
Media	40.200000	1289.600000	0.000000	197.800000

(b) Infeasibility

	Iris	Ecoli	Rand	Tyroid
Semilla 37	0.936712	47.200253	0.715599	20.779371
Semilla 42	0.874824	47.616898	0.715599	20.961143
Semilla 440	0.903969	47.883408	0.715599	20.719147
Semilla 699	0.973378	47.606503	0.715599	21.181446
Semilla 752	0.920325	48.012620	0.715599	20.775734
Media	0.921842	47.663936	0.715599	20.883368

(c) Función objetivo

	Iris	Ecoli	Rand	Tyroid
Semilla 37	24.114498	249.527510	24.124850	45.735350
Semilla 42	24.452518	243.258400	23.513433	44.398865
Semilla 440	23.470268	241.543440	23.613468	44.316753
Semilla 699	23.512020	241.600170	23.420820	44.240932
Semilla 752	23.493227	241.310170	23.466751	44.134445
Media	23.808506	243.447938	23.627864	44.565269

(d) Tiempo de ejecución

Figura 1: Resultados para woa con 10 % de restricciones

	Iris	Ecoli	Rand	Tyroid
Semilla 37	0.684892	12.979752	0.715599	13.340351
Semilla 42	0.663448	13.042139	0.715599	13.592555
Semilla 440	0.662933	13.033968	0.715599	13.620830
Semilla 699	0.660625	13.035360	0.715599	13.909627
Semilla 752	0.658393	13.160732	0.715599	13.623517
Media	0.666058	13.050390	0.715599	13.617376

(a) Distancia intracluster

	Iris	Ecoli	Rand	Tyroid
Semilla 37	77.000000	2637.000000	0.000000	371.000000
Semilla 42	55.000000	2553.000000	0.000000	294.000000
Semilla 440	81.000000	2663.000000	0.000000	426.000000
Semilla 699	76.000000	2512.000000	0.000000	337.000000
Semilla 752	90.000000	2625.000000	0.000000	348.000000
Media	75.800000	2598.000000	0.000000	355.200000

(b) Infeasibility

	Iris	Ecoli	Rand	Tyroid
Semilla 37	0.928991	48.354507	0.715599	20.122843
Semilla 42	0.837804	47.290054	0.715599	18.967363
Semilla 440	0.919712	48.757507	0.715599	21.408813
Semilla 699	0.901553	46.733270	0.715599	20.070543
Semilla 752	0.943702	48.374510	0.715599	19.985538
Media	0.906353	47.901970	0.715599	20.111020

(c) Función objetivo

	Iris	Ecoli	Rand	Tyroid
Semilla 37	28.139378	293.534940	28.877132	53.587063
Semilla 42	28.537369	284.490780	27.549004	52.060455
Semilla 440	27.767319	283.893800	27.526875	52.108093
Semilla 699	27.650816	283.500640	27.543203	52.094227
Semilla 752	27.724705	282.832120	27.512619	51.931313
Media	27.963917	285.650456	27.801767	52.356230

(d) Tiempo de ejecución

Figura 2: Resultados para woa con 20 % de restricciones

4. Análisis de los resultados

En primer lugar vamos a proceder a comparar los resultados obtenidos con los resultados usando otros algoritmos

	Iris			Ecoli			Rand			Newthyroid		
	tasa C	inf	score	tasa C	inf	score	tasa C	inf	score	tasa C	inf	score
greedy	1.800	355.8	4.057	44.613	1500.4	84.868	2.227	299.4	4.383	13.160	958.2	48.203
ls	0.669	0.000	0.669	21.604	92.200	24.078	0.716	0.000	0.716	13.246	24.4	14.138
es	0.669	0.000	0.669	21.679	83.800	23.927	0.716	0.000	0.716	12.628	47.400	14.362
bmb	0.669	0.000	0.669	21.997	156.000	26.183	0.716	0.000	0.716	13.835	6.000	14.054
ils	0.669	0.000	0.669	21.781	75.800	23.815	0.716	0.000	0.716	13.835	6.000	14.054
ils-es	0.669	0.000	0.669	28.198	332.800	37.127	0.716	0.000	0.716	13.835	6.000	14.054
woa	0.666	75.800	0.906	13.050	2598.000	47.902	0.716	0.000	0.716	13.617	355.200	20.111

Cuadro 1: Resultados medios obtenidos para el problema del PAR con restricciones del 10 %