Escuela Técnica Superior de Ingeniería Informática.
ULL. Miguel Pérez Bello. Alu4117.

2014

Informe Computacional

mayo 2

2014

Conjunto de algoritmos y técnicas aplicadas sobre los mismos para resolver el problema del agrupamiento o Clustering.

Práctica de Diseño y Análisis de Algoritmos.

I. Estructuras de datos y diseño general:

Dado que el problema al que nos enfrentamos consiste en agrupar una serie de objetos con características (en nuestro caso, numéricas) en un conjunto de grupos o clusters, se ha realizado un diseño de elementos o componentes que modelan a los mismos:

1. Se dispone de una clase Objeto, que posee unas dimensiones (número de características), un vector de características y almacena el número o identificador del cluster en que se clasificó.

2. Se dispone además de una estructura o clase Cluster, que posee además de las dimensiones y las características, un identificador único para realizar el agrupamiento. Un cluster debe distar lo más posible de otro cluster en cuanto a características, pero los objetos de un mismo cluster, entre sí, deben parecerse lo más posible.

II. Algoritmos implementados y submétodos implicados:

Para la implementación de los algoritmos de clustering, se ha optado por un diseño basado en el patrón de diseño software denominado "patrón Strategy o patrón estrategia". Dicho patrón se ha utilizado de la siguiente forma:

1. Se posee una clase base "AlgoritmoClustering" que posee las funcionalidades comunes a cualquiera de los algoritmos de agrupamiento implementados.

2. Se tiene un objeto "Simulador" que actúa de interfaz de usuario, permitiéndole a éste la oportunidad de establecer las instancias para la ejecución, el tipo de algoritmo a ejecutar, etc...

3. Para la implantación del patrón estrategia, en la clase Simulador se posee un puntero a un tipo de algoritmo base "AlgoritmoClustering" esto permite, a través de polimorfismo, establecer con qué estrategia (algoritmo) afrontar la resolución del problema (realizar el agrupamiento).

4. Por todo lo comentado, queda implícito que se han desarrollado varias clases, una por algoritmo de agrupamiento, como se muestra en el siguiente diseño:

+ AlgoritmoClustering : Clase base de la que heredan:

- Kmedias.
- AlgVoraz.
- AlgGRASP.
- TecnicaMultiarranque.
- TecnicaVNS.

1. Kmedias:

Este algoritmo es uno de los más sencillos para realizar agrupamientos. El método consiste básicamente en generar una serie de **k** centroides (clusters), asignar cada objeto a su centroide más cercano, recalcular los centroides iniciales y repetir el proceso citado mientras haya cambios de asignación.

Un pseudocódigo podría ser el siguiente:

```
generamosCentroidesIniciales ( );
hacer:
  begin hacer
        asignarACentroideMasCercano ( );
        recalcularCentroides ( );
mientras (cambie algún objeto de cluster);
  fin hacer.
```

Notas:

- La generación de los centroides iniciales implementada permite al usuario elegir el número de los **k** clusters y si se generan de forma aleatoria o leyendo los datos desde un fichero de entrada de texto plano.
- Recalcular los centroides promedia las características de los objetos pertenecientes a cada cluster y asigna dicho promedio al cluster en cuestión.

Lo bueno que tiene este algoritmo es que, si inicialmente los centroides generados no son muy representativos, por el cálculo propio de las k medias, estos centroides finalizan ajustándose mucho más a los apropiados para el agrupamiento deseado.

2. Algoritmo Voraz (Greedy):

Al pensar en la implementación de una técnica que, además de tener la cualidad de voraz (elegir en cada paso la mejor opción), tendría que ser constructivo, es decir, para el caso al que nos enfrentamos, debería ir generando centroides iniciales para el agrupamiento donde mejor convenga a las instancias de entrada en cada caso.

Para ello se ha optado por la siguiente forma de generar los centroides iniciales:

1º Se toma uno de los objetos al azar como primer centroide.
2º Para los k-1 objetos restantes:
- Se asigna como nuevo centroide el objeto más alejado a los centroides que ya han sido asignados en ese momento. Es decir, se suma la distancia de todos los objetos a todos los clusters existentes en ese momento y se toma como nuevo cluster el objeto que presente un sumatorio mayor de distancias.

La fase final del algoritmo la constituye un proceso de ir asignando cada objeto a su cluster más cercano. Para ello se elabora una tabla de distancias objetos-clusters y se va tomando en cada iteración la de menor distancia, ese dato nos dice qué objeto será asignado a qué cluster. Se continúa el proceso mientras no se tenga una solución completa del problema.

3. Técnica GRASP:

La técnica GRASP, denominada Greedy Randomnized Adaptative Search Procedure, se basa en la idea intuitiva de que una técnica voraz, al intentar en cada paso ir a por la mejor opción para alcanzar la mejor solución, se estanca en soluciones que son buenas, pero no son las mejores ya que a veces hay que tomar una de las mejores decisiones en el camino y no la mejor absoluta, para llegar a la mejor solución de todas.

Esta idea parece un poco bestia si tratamos de imaginarnos que el hecho de añadir un componente aleatorio a una búsqueda va a conducir a mejores soluciones, pero en la práctica así sucede porque, puede que a la primera no se obtenga una mejor solución que la técnica voraz, pero si se ejecuta múltiples veces esta técnica, podremos ver como en varias ocasiones se mejoran las soluciones.

El procedimiento actúa de la siguiente manera:

1º Se generan **n** centroides de forma aleatoria.
2º Se elabora una tabla que representa las distancias existentes entre cada objeto y cada cluster.
3º Se elabora una Lista Restringida de Candidatos (LRC) cuyo tamaño lo define el usuario y en la que se incluyen los **m** objetos que menor distancia presentan con alguno de los clusters.
3º De los **m** mejores candidatos de la LRC se escoge uno al azar y se asigna como parte de la solución final.
4º El proceso se repite hasta que se obtenga una solución completa del problema.

4. Técnica MultiArranque:

Este tipo de técnica, como su propio nombre indica, tienen la peculiaridad de realizar múltiples arranques a técnicas locales que generan una solución del problema (pudiendo estancarse en un óptimo local), con el objetivo de ir comparando soluciones desde varios puntos de inicio e ir quedándose con la solución de mejor valor objetivo.

Un pseudocódigo podría ser el siguiente:

```
solución actual = busquedaLocal (centroides al azar);
mejor solución = solución actual;
contadorMejora = 0;

hacer:
    si (valorObjetivo (solución actual) > valorObjetivo (mejor solución):
        mejor solución = solución actual;
        contadorMejora = 0;
    Sino:
        contadorMejora ++;
    finSiNo.

    solución actual = busquedaLocal (centroides al azar);

mientras (no se mejore durante x veces la solución: contadorMejora < X);
finHacer.
```

Eligiendo el número de movimiento de no mejora permitidos adecuado y aplicando una buena búsqueda local, se pueden obtener muy buenos resultados con esta técnica.

En este caso, la técnica se ha implementado dándole al usuario la opción de si utilizar como búsqueda local un algoritmo greedy o una técnica GRASP.

5. Técnica VNS (Búsqueda por Entorno Variable):

El tipo de técnica que realiza movimientos por vecindades se basa en la idea de ir recorriendo el espacio de soluciones vecinas a una solución inicial con el objetivo de ir quedándose con la mejor de sus vecinas en cada caso.

El concepto de vecindad en este caso hace referencia a que dos soluciones son vecinas si no distan, en cierta unidad de distancia, los resultados de las mismas.

Para el caso concreto del clustering, se ha optado dar la opción al usuario de elegir el nivel de vecindad máximo entre soluciones. Este nivel indica el número de elementos de la solución que serán mutados, como máximo.

Una solución se codifica como un vector en el que cada posición ***i*** del mismo representa un objeto del problema (indexados e la misma forma) y cada valor **i** del vector representa el cluster al que se ha asignado el objeto en cuestión. Por ello, para "mutar" una solución a una de sus vecinas, se realiza un procedimiento de cambio aleatorio de **n** objetos de cluster. De este modo, un pseudocódigo de lo descrito podría ser el siguiente:

```
solucionActual = busquedaLocal ( );
mejorSolucio = solucionActual;
desde k = 1 a k máximo hacer:
        solucionActual = mutar (solucionActual, k unidades);
        si (Objetivo (solucionActual) > Objetivo (mejorSolucion)) entonces:
                mejorSolución = solucionActual;
                k = 1;
        sino hacer:
                k++;
        finSiNo.
FinHacer.
```

El pseudocódigo mostrado es muy general, pero aplicado a nuestro caso, se genera una solución inicial a partir del uso de una técnica GRASP y se va mutando dicha solución por sus vecinas mientras vaya habiendo mejoras y no se llegue al máximo en vecindad.

Al finalizar el procedimiento, nos quedamos con la primera mejor de todas las vecinas a nuestra solución, porque en cada caso cambiamos a la mejor vecina y seguimos mutando hasta llegar a un máximo de vecindad sin mejora.

III. Tablas de resultados Experimentales:

1. Resultados del Algoritmo Kmedias:

Problema	m	k	SSE	CPU (ms)
Kmedias	2	3	293.125	0.0590086
Kmedias	2	4	188.484	0.0782013
Kmedias	2	5	142.264	0.100115
Kmedias	2	6	138.462	0.101848
Kmedias	3	3	599.167	0.05095
Kmedias	3	4	465.716	0.56784
Kmedias	3	5	367.021	0.112322
Kmedias	3	6	282.567	0.12908
Kmedias	4	3	1163.86	0.104904
Kmedias	4	4	964.476	0.14441
Kmedias	4	5	709.227	0.2265
Kmedias	4	6	697.011	0.259089
Kmedias	5	3	1295.88	0.106122
Kmedias	5	4	951.1	0.153494
Kmedias	5	5	861.579	0.189097
Kmedias	5	6	757.201	0.198214

Nota:

m: número de características

K: número de clusters.

SSE: Error del agrupamiento final.

CPU: tiempo de CPU consumido.

2. Resultados de la Técnica Voraz:

Problema	**m**	**k**	**SSE**	**CPU (ms)**
Voraz	2	3	437,305	0,07301
Voraz	2	4	405,619	0,081566
Voraz	2	5	643,288	0,097036
Voraz	2	6	780,398	0,977516
Voraz	3	3	1331,65	0,213623
Voraz	3	4	927,695	0,245571
Voraz	3	5	781,298	0,303507
Voraz	3	6	720,383	0,36786
Voraz	4	3	2271,1	0,2117116
Voraz	4	4	2254,81	0,147581
Voraz	4	5	2079,04	0,102043
Voraz	4	6	1983,56	0,2003371
Voraz	5	3	2409,55	0,274897
Voraz	5	4	2141,62	0,35426
Voraz	5	5	1164,5	0,452995
Voraz	5	6	1043,48	0,53215

Nota:

m: número de características

K: número de clusters.

SSE: Error del agrupamiento final.

CPU: tiempo de CPU consumido.

3. Resultados de la Técnica GRASP:

Problema	**m**	**k**	**|LRC|**	**SSE**	**CPU (ms)**
GRASP	2	4	2	1107,99	0,182629
GRASP	2	4	3	1128,04	0,167131
GRASP	2	5	2	1107,98	0,132322
GRASP	2	5	3	1250,39	0,149965
GRASP	3	4	2	1078,58	0,386
GRASP	3	4	3	1149,33	0,454664
GRASP	3	5	2	1022,52	0,327826
GRASP	3	5	3	1114,25	0,32711
GRASP	4	4	2	2266,27	0,313997
GRASP	4	4	3	1736,43	0,382423
GRASP	4	5	2	1171,24	0,394106
GRASP	4	5	3	2635,95	0,473996
GRASP	5	4	2	2648	0,379086
GRASP	5	4	3	2630,59	0,386238
GRASP	5	5	2	2281,02	0,54121
GRASP	5	5	3	1795,96	0,498295

Nota:

m: número de características

K: número de clusters.

|LRC|: tamaño de la **L**ista **R**estringida de **C**andidatos.

SSE: Error del agrupamiento final.

CPU: tiempo de CPU consumido.

4. Resultados de la Técnica MultiArranque:

Problema	**m**	**k**	**noM**	**SSE**	**CPU (ms)**
MultiArranque	2	3	100	266	1,05031
MultiArranque	2	3	500	352.537	105,947
MultiArranque	2	4	1000	267,826	108,44
MultiArranque	2	4	5000	113,81	369,694
MultiArranque	3	4	100	651,265	1,96651
MultiArranque	3	5	500	539,292	196,036
MultiArranque	3	3	1000	688,71	201,063
MultiArranque	3	4	5000	483,4	500,67
MultiArranque	4	5	100	951,63	1,03184
MultiArranque	4	3	500	840,92	110,148
MultiArranque	4	4	1000	824,75	170,169
MultiArranque	4	5	5000	603,06	900,37
MultiArranque	5	3	100	493,027	2,04965
MultiArranque	5	4	500	1006,41	108,414
MultiArranque	5	5	1000	946,81	198,327
MultiArranque	5	5	5000	931,4	898,49

Nota:

m: número de características

K: número de clusters.

nM: número de movimientos **de no mejora** permitidos. IMPORTANTE!

SSE: Error del agrupamiento final.

CPU: tiempo de CPU consumido.

|LRC|: el tamaño de la **L**ista **R**estringida de **C**andidatos (siempre 2).

Aunque se haya implementado la técnica multiarranque con una técnica voraz y otra grasp, los resultados que se muestran en la tabla se han extraído de aplicar el multiarranque a la técnica grasp.

5. Resultados de la Búsqueda por Entorno Variable (VNS):

Para la generación de la solución inicial se ha optado por emplear la técnica GRASP (con |LRC| = 2).

A partir de la solución inicial generada con el GRASP, se va mutando en **k** unidades (hasta **kMax**) y vamos almacenando aquella solución que mejore la actual.

Una vez no se mejora en la vecindad y se ha llegado a los límites de la misma (k = kMax), se finaliza la ejecución quedándonos con la mejor de todas las soluciones obtenidas por el camino.

Problema	**m**	**k**	**kMax**	**SSE**	**CPU (ms)**
VNS	2	3	10	789,017	0,20957
VNS	2	3	100	407,026	0,767231
VNS	2	4	500	631,178	2,19059
VNS	2	4	1000	593,695	5,13268
VNS	3	3	10	1461,91	0,368118
VNS	3	3	50	1005,93	0,420809
VNS	3	4	100	1772,45	0,778913
VNS	3	4	500	1442,53	2,02012
VNS	4	3	10	2363,35	0,304699
VNS	4	3	50	2169,23	0,674248
VNS	4	4	100	1672,83	0,945091
VNS	4	4	500	1251,12	2,03633
VNS	5	3	100	1873,45	0,989676
VNS	5	3	500	1082,62	2,27094
VNS	5	4	1000	2317,53	5,39899
VNS	5	4	5000	1948,96	9,93794

Nota:

m: número de características

K: número de clusters.

kMAx: número que permite definir una estructura de entorno. Una solución es vecina si no difiere en kMax unidades de la solución actual, es decir, si solo posee kMax objetos que difieren en el cluster al que pertenecen.

SSE: Error del agrupamiento final.

CPU: tiempo de CPU consumido.

6. Conclusiones:

Más que exponer solo cuál de todas las técnicas es la que obtuvo los mejores resultados, se expondrá qué es lo mejor y peor de cada uno de los algoritmos de agrupamiento implementados.

Comenzando por la técnica más simple, que es el kMedias, cabe comentar que resulta certera a la hora de ir ajustando los centroides de los clusters a base de promediar las características de cada objeto. El inconveniente es que se queda estancada la búsqueda en óptimos locales debido a la aleatoriedad de la generación de los centroides iniciales, entre otras,…

La aleatoriedad al generar los centroides iniciales que tiene el kMedias, la solventa en parte la técnica greedy o voraz al ir colocando los centroides basándose en la propia definición de cluster. Puesto que un cluster es un grupo cuyas características difieren lo máximo posible con las características de otros clusters, se ha optado por generar los centroides de la forma: primer centroide aleatorio y el resto irlo colocando lo más alejado posible de los centroides existentes en el momento.

Lo peor de la técnica voraz es que se base en ir eligiendo en cada momento aquella opción que presente un mejor valor objetivo (parcial) porque esto conducirá al mejor valor objetivo posible para la solución global. Este fundamento no es del todo erróneo, pero también provoca que la búsqueda quede estancada en óptimos locales.

Añadiendo un componente de aleatoriedad a esa persecución del mejor en cada paso, conseguimos en algunos casos que, mejoren las soluciones globales, por el hecho de empeorar de vez en cuando la mejor opción parcial, este componente aleatorio lo posee la técnica GRASP.

Lo que ocurre con las técnicas GRASP es que para que se consiga mejorar una técnica voraz, debería realizarse múltiples ejecuciones de la misma e ir guardando la mejor de las soluciones en cada caso. Hecho que se ve reflejado en la implementación de la técnica MultiArranque. Dicha técnica consigue que el GRASP obtenga mejores resultados en la mayoría de casos que la técnica Voraz.

La última técnica implementada fue la Búsqueda por Entorno Variable, de la que comentar que, al obtenerse una solución buena, pero no la mejor con una técnica GRASP (inicialmente), al mutar entre soluciones vecinas (de forma aleatoria) conseguimos, en ocasiones, que la técnica solvente los errores de las elecciones aleatorias hechas por el GRASP.

Por tanto y como se demuestra en las tablas expuestas en este documento, las técnicas VNS y MultiArranque son las que se llevan las medallas de Oro y Plata debido a que son las que presentan los errores en el agrupamiento (SSE) más bajos. Entre ellas, la elección de cuál es la mejor queda a elección del usuario y del contexto en el que se quieran implementar debido a que, la técnica MultiArranque obtiene muy buenos resultados, pero a base de ejecutar múltiples veces la búsqueda local, lo que conlleva un consumo en tiempo a tener en cuenta. Por otra parte, la técnica VNS obtiene resultados muy buenos, aunque un poquito peor que la MultiArranque, pero mejora a esta en consumo de tiempo.