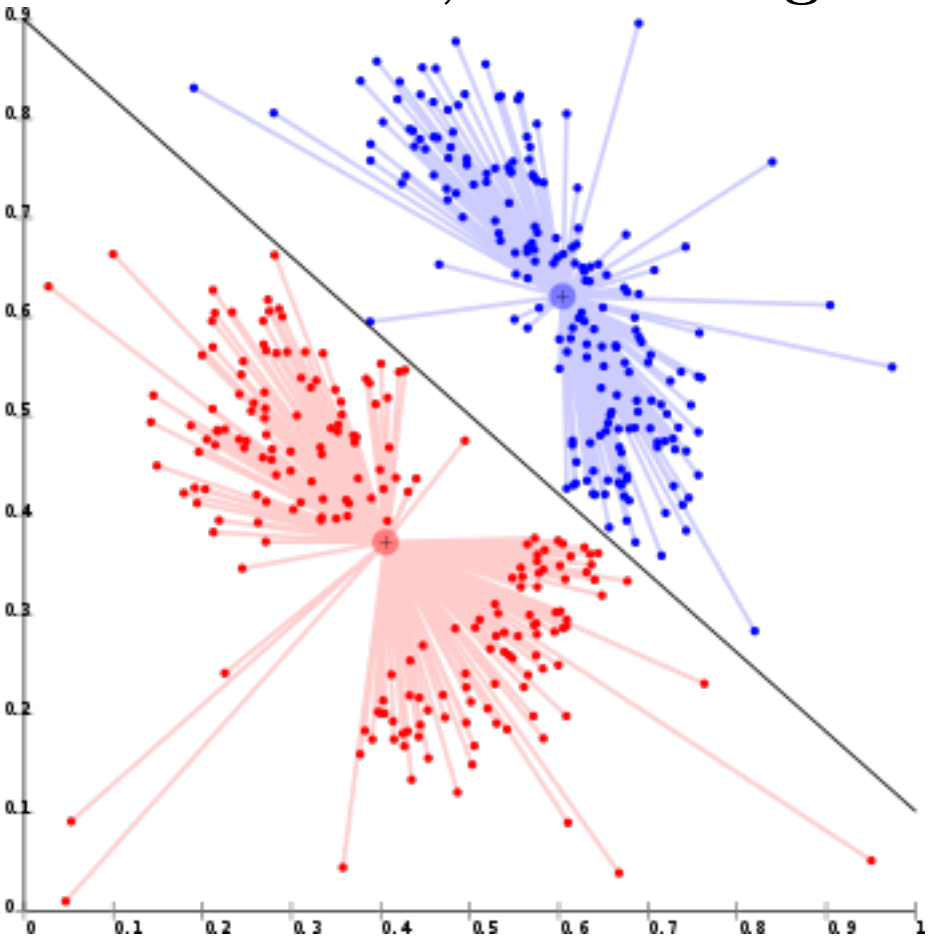
Práctica 6, Clustering



Minería de datos

Andoni Martín Reboredo
David Ramirez Ambrosi

26 de octubre de 2014

Índice general

Índice de figuras

Capítulo 1

Introducción

El presente documento constituye el resultado de la práctica realizada en base a la implementación del algoritmo de clasificación no supervisada **K-Means clustering**. Este algoritmo trata el agrupamiento de un conjunto de instancias en base a su proximidad con las demás instancias contenidas en el espacio de muestra proporcionado al algoritmo para su ejecución.

Dentro del algoritmo cabe el estudio de diferentes variaciones en los distintos parámetros de que dispone. Nosotros hemos considerado variaciones sobre dos parámetros, el método de cálculo de la distancia entre los distintos elementos que posee el cluster y la inicialización de los distintos clusters. Esta inicialización servirá como base de las sucesivas iteraciones que conforman el algoritmo.

1.1. Clasificación no-supervisada

La clasificación no supervisada es aquella que se lleva a cabo mediante el estudio de las diversas instancias que conforman el espacio de aplicación del algoritmo sin que estas instancias tengan que estar previamente clasificadas dentro de una clase [1].

Se trata de una técnica de exploración de los datos en la que se intentan detectar estas clases desconocidas. Dependiendo de el algoritmo de clasificación utilizado, el número de clases debe o no ser especificado. Por ejemplo, en el algoritmo en que se basa este trabajo debe ser especificado, sin embargo en técnicas de **clusterig jerárquico** no.

1.2. Objetivo

Esta práctica tiene como objetivo principal la comprensión de los procesos internos que realiza un algoritmo de agrupamiento cualquiera como puede ser el K-Means clustering. El aprendizaje se realizará de forma práctica a través de la implementación del algoritmo K-Means clustering junto con diversas opciones con las que realizar algunos pasos del mismo, como son el uso de métricas o inicializaciones del algoritmo diferentes. Estas variaciones requieren que el algoritmo sea entendido plenamente para poder hacer contribuciones que tengan utilidad para la realización del proceso.

Capítulo 2

Algoritmo

2.1. K-means, algoritmo principal

```
inicializar
divergencia = infinito

Mientras(numiteraciones <= iteracionesIndicadas AND divergencia < delta)
{
    centroides = centroidesNuevos

    calcularPertenencias
    centroidesNuevos = calcularNuevosCentroides

    calcularDivergencia(centroidesNuevos)
}
```

2.2. Subrutina inicialización

2.2.1. Inicialización aleatoria

```
Para cada dimensión
{
   mientras extraiga una instancia ya evaluada
   {
      extraigo una instancia nueva
   }
   añado la instancia extraida a las evaluadas
   establezco la instancia como centroide de un cluster
}
```

2.2.2. Pertenencia aleatoria

```
Mientras haya instancias que asignar
```

```
{
   Calculo un número de cluster aleatorio
   Extraigo la siguiente instancia
   Añado la instancia al cluster aleatorio
}
Calculo los centroides del cluster
```

2.2.3. División de espacio

```
Obtengo los rangos máximos y mínimos de cada subespacio
Mientras no haya creado k divisiones
{
   Mientras no haya establecido todos los atributos(recorrido los subespacios)
   {
      Divido el subespacio en K
      Asigno el centro del subespacio dividido correspondiente al índice del bucle
   }
   Añado el centroide resultante de dividir el espacio
}
```

2.2.4. Generación aleatoria de codewords

```
Obtengo los máximos y mínimos de cada dimensión
Para cada cluster
{
   Evaluo cada dimensión
   {
      Calculo un valor aleatorio para ese centroide en esa dimensión
   }
   Añado el centroide generado al cluster
}
```

2.3. Subrutina calcularPertenencias

```
Crear nuevos clusters

Para cada instancia
{
    Para cada centroide
    {
        distancia entre la instancia y cada cluster
        Guardamos los mínimos
    }
    Para cada centroide obtenido
        Guardamos la instancia en el cluster correspondiente al centroide
}
```

2.4. Subrutina calcularCentroides

```
Para cada cluster

   Calcular la instancia media

return nuevosCentroides
```

2.5. Subrutina calcularDivergencia

```
Para cada cluster

   Calcular la distancia entre el centroide antiguo y el nuevo

return divergenciaAcumulada
```

Capítulo 3

Diseño

En la figura 3.1 se muestra el diagrama de clases final de la aplicación, extraído a partir del código mediante el software *Visual Paradigm*. La especificación de cada uno de los métodos se puede encontrar en la documentación adjunta generada con *Eclipse*, la carpeta *doc* contiene la *javadoc* generada en forma de *HTMLs* navegables.

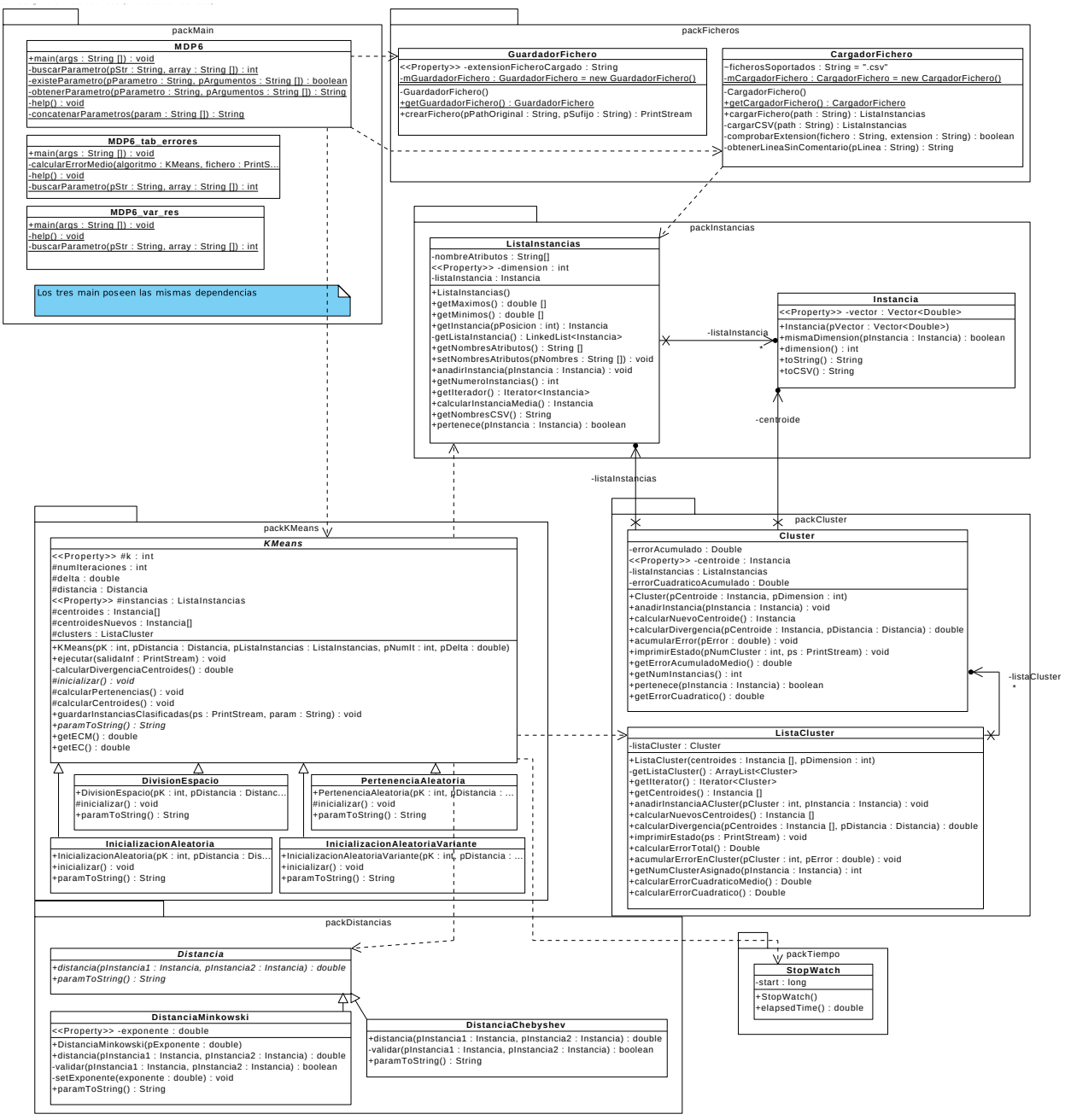


Figura 3.1: Diagrama de clases y paquetes

Capítulo 4

Resultados experimentales

4.1. Banco de pruebas para la validación de software y resultados

Para la validación de nuestro software, MDP6, utilizaremos el software Weka usado en la asignatura. Ambos softwares, el propio y Weka, difieren en capacidad y soporte de funciones. Para la realización de pruebas, describimos las capacidades de cada uno de ellos:

MDP6

- **Inicializaciones**. Disponemos de 4 tipos de inicializaciones diferentes:
  - **Inicialización aleatoria**. Elige al azar k instancias del conjunto de datos para actuar de centroides.
  - **Inicialización por petenencia aleatoria**. Por cada instancia, la incluye al azar en uno de los k clusters. Posteriormente se calculan los centroides.
  - **Inicialización por generación aleatoria de centroides**. Se recogen los rangos en que varían los atributos de todas las instancias. Posteriormente se crean k centroides con valores al azar incluidos en los diferentes intervalos identificados.
  - **Inicialización por división de espacio**. Se divide el espacio muestral de acuerdo al número de clusters a crear y se identifican los centroides.
- **Distancias**. Se han implementado dos tipos de métricas capaces de trabajar con el algoritmo
  - **Distancia Minkowski**. Implementada para cualquier valor del parámetro m (siempre que sea un real positivo mayor o igual de 1).
  - **Distancia Chebyshev**.
- **Criterio de parada**. Se ofrece la posibilidad de establecer un límite de iteraciones para la ejecución del algoritmo o bien el uso de un valor a satisfacer para la divergencia entre conjuntos de centroides de dos iteraciones diferentes. También se pueden combinar ambos valores de forma que el primero en satisfacerse determine el fin del algoritmo.
- **Tipos de atributos**. Únicamente se permiten valores numéricos y conocidos para los atributos de las instancias. Todo fichero que vaya a ser cargado debe satisfacer esta condición. No se ofrece funcionalidad como la normalización de atributos.

- **Medidas de calidad**. Como medida del error tras el proceso de clustering se ofrece tanto el **Error Cuadrático** como el **Error Cuadrático Medio**.
- **Clasificación de instancias**. Se puede generar un archivo conteniendo las instancias cargadas (en el mismo orden al de entrada) en formato CSV. En este fichero de salida, a cada instancia se le ha añadido un atributo que indica el cluster al que finalmente fue asociada tras la ejecución del algoritmo.
- **Repetición de pruebas**. Como se explicará posteriormente, las pruebas realizadas con tipos de inicializaciones que utilicen generación de valores aleatorios no pueden ser repetidas ya que los valores variarán en cada ejecución.

Weka

- **Inicializaciones**. No ofrece más que un tipo de inicialización.
- **Distancias**. Las distancias disponibles para el algoritmo son Minkowski 1 y Minkowski 2 (Euclídea y Manhattan respectivamente). Además, cada una de estas dos distancias permite normalizarlas.
- **Criterio de parada**. Permite establecer el número máximo de iteraciones que realizará el algoritmo.
- **Tipos de atributos**. Soporta nominales, numéricos, binarios, unarios, con valores desconocidos o nominales vacíos.
- **Medidas de calidad**. En el caso de la distancia **Euclídea**, se ofrece el **Error Cuadrático** como medida de error. En el caso de la distancia **Manhathan**, la medida ofrecida es la **suma de las distancias dentro del cluster**.
- **Clasificación de instancias**. Tras ejecutar el algoritmo, se puede visualizar el resultado de la agrupación de las instancias y guardarlas en formato **ARFF** con un atributo más correspondiente al cluster con que la instancia ha sido asociada.
- **Repetición de pruebas**. Weka permite que para una misma semilla (parámetro configurable) los valores generados carezcan de variación, de forma que se puede repetir el mismo experimento una y otra vez para una misma semilla.

Los ficheros utilizados como fuente de instancias han sido **previamente procesados** con *Weka* para adaptarlos a nuestro software. Se ha procedido a **eliminar el atributo clase** en aquellos ficheros que la tuvieran y se han guardado las instancias en **formato CVS**.

Así mismo, en algunos casos ha sido necesario utilizar además un **procesador de texto** para poder obtener los resultados deseados. También se ha usado el software **R** para obtener ciertos resultados estadísticos y realizar comparaciones de valores. Para automatizar el proceso de obtención de datos, se han desarrollado programas **JAVA** extra y se ha utilizado un script del interprete **BASH**.

Para la representación de resultados se ha utilizado el software **LibreOffice Calc** y la extensión **calc2latex** para generar el código Latex de las tablas presentes en este informe.

4.1.1. Pruebas realizadas

Se han realizado tres tipos diferentes de pruebas para comprobar la validez y rendimiento del software desarrollado. En cada una de estas pruebas los indicadores de referencia utilizados han sido:

- **Medidas de error**. Comparación directa del error generado en el proceso de clustering. Prueba desarrollada en el apartado 4.2, se utiliza el **Error Cuadrático** para realizar comparar directamente el software con Weka.
- **Medidas de precisión**. Comparación de la **precisión** capaz de alcanzar nuestro software con la que Weka es capaz de alcanzar. El proceso se desarrolla y explica en el apartado 4.3.
- **Medidas estadísticas**. Uso de medidas como máximo, mínimo, media y varianza de un conjunto de datos para determinar la variabilidad de las repeticiones sucesivas de un determinado experimento con nuestro software. Proceso desarrollado en el apartado 4.4.

4.2. Resultados comparativos

En este apartado comparamos el software implementado con el algoritmo Kmeans de Weka por medio del error cuadrático. Este indicador de error solo se muestra en Weka al utilizar la distancia Euclídea, por lo tanto las comparaciones se limitan a este tipo de distancia. Además, nuestro software no es capaz de normalizar las instancias, así que desactivaremos esta opción al trabajar con Weka.

En cuanto al resto de parámetros, para cada uno de los ficheros de pruebas se ha variado el parámetro **K** en el conjunto {2,3,5,7,9}. Por otro lado, el tipo de inicialización en Weka es único, pero con nuestro software hemos utilizado las 4 posibles.

Otro dato a tener en cuenta es la gran dificultad de repetir un mismo resultado, para una misma configuración de parámetros, en nuestro software por el uso de aleatorios. Esto genera el problema de comparar el error variable de nuestro software con el de Weka.

Para solucionarlo, hemos hecho 100 ejecuciones de cada combinación de parámetros posible y hemos utilizado la media de los diferentes errores obtenidos para conseguir un valor más estable del error cometido por nuestro software. El proceso con el ejecutable estándar desarrollado sería muy costoso, así que hemos desarrollado el programa *MDP6_tab_errores.jar* para automatizar dicho proceso.

Los ficheros correspondientes a este apartado están disponibles en la carpeta */resultados/resultados_comparativos*. Se encuentran los ficheros originales, los ficheros operativos y los fichero resultado obtenidos. El resumen de los resultados se encuentra a continuación.

Error Cuadrático	Inic_aleatoria	Pertenencia_aleatoria	Cent_aleatorios	División_espacio	Weka
K = 2	17.022.812.094,779	16.662.529.293,188	16.662.595.771,055	16.662.529.293,188	16.662.529.293,188
K = 3	15.367.892.995,782	15.308.965.075,088	15.276.313.379,049	15.068.933.772,772	15.347.180.933,720
K = 5	13.194.207.465,418	13.100.202.144,746	13.055.014.852,916	14.786.712.947,628	13.128.804.126,107
K = 7	11.748.015.592,402	11.590.997.272,742	11.685.296.452,310	13.886.614.328,470	12.208.057.266,416
K = 9	10.626.841.728,292	10.711.671.635,239	10.678.692.422,634	13.537.274.017,576	11.151.380.874,620

Figura 4.1: Error cuadrático para el fichero colon.arff. Distancia Euclídea

Error Cuadrático	Inic_aleatoria	Pertenencia_aleatoria	Cent_aleatorios	División_espacio	Weka
K = 2	218.924,945	209.417,829	210.407,601	202.489,423	202.489,423
K = 3	119.441,033	108.167,063	107.651,418	110.504,153	110.504,153
K = 5	55.416,837	52.625,441	52.134,900	36.139,549	35.296,151
K = 7	36.428,538	39.363,275	40.904,587	35.296,151	15.905,814
K = 9	23.140,147	29.075,157	31.384,815	24.104,840	10.600,845

Figura 4.2: Error cuadrático para el fichero food.arff. Distancia Euclídea

Error Cuadrático	Inic_aleatoria	Pertenencia_aleatoria	Cent_aleatorios	División_espacio	Weka
K = 2	152,369	152,369	152,369	152,369	152,369
K = 3	92,647	96,026	91,888	145,279	78,941
K = 5	52,882	58,886	60,564	680,824	53,007
K = 7	41,422	51,064	51,155	680,824	38,926
K = 9	33,432	46,544	45,872	680,824	28,474

Figura 4.3: Error cuadrático para el fichero iris.arff. Distancia Euclídea

4.3. Clasificación supervisada respecto de otro software de referencia

En este apartado observamos el cluster asignado a cada instancia de un fichero por ambos softwares comparándolo con la clase que la instancia tenía en el fichero original. El objetivo es obtener la tasa de aciertos que los diferentes softwares son capaces de alcanzar.

En este caso no se han tenido en cuenta la variabilidad de los resultados de nuestro software y se ha comparado el de una sola ejecución.

En cuanto a parámetros, se han tenido en cuanta las diferentes inicializaciones de nuestro software y varias distancias de las soportadas. En Weka se han utilizado las dos distancias soportadas (Manhattan y Euclídea) y por cada una se ha probado con y sin normalización.En cuanto al parámetro K, se establece igual al número de clases diferentes del fichero original. Estas pruebas se han basado en el fichero *iris.arff*, por lo que el valor de K usado ha sido **3**.

Para obtener las instancias etiquetadas en nuestro software se ha utilizado el ejecutable *MDP6.jar* que automáticamente genera un fichero conteniendo las diferentes instancias y su cluster correspondiente. La variación de parámetro se ha llevado a cabo mediante un script BASH, *MDP6_param_variator.sh*, con el que hemos obtenido los diferentes ficheros resultados.

En el caso de Weka, el proceso se ha realizado de forma manual mediante su interfaz, ya que eran solo 4 las diferentes combinaciones de parámetros a probar.

Para realizar la comparación con el fichero original de instancias ha sido preciso reetiquetar los valores de la clase estimados de forma que las clases se correspondieran en la mejor medida posible al fichero original. El fichero original también ha sido procesado para la comparación, variando el formato a CSV y cambiando el valor nominal de las clases por un valor numérico.

Los ficheros resultado de este experimento se encuentran en */resultados/clasificación_supervisada*, siguiendo la misma estructuración que los del apartado anterior.

En este caso los resultados son bastante similares los unos a otros, la mayoría de casos fallan 17 instancias de clasificación, un 8.6 %. La única ejecución que obtiene una mayor tasa de aciertos es la realizada mediante centroides aleatorios para tres clusteres y sobre distancia euclídea que resulta en solo 16 fallos. Por lo demás, existe también un gran grupo de resultados que aciertan 100 instancias nada más, cuestión que puede deberse a la limitación de iteraciones o a un mal resultado en la selección aleatoria de centroides iniciales que resultan en una convergencia no deseable.

Software	Inicialización	Distancia	Norm	% aciertos	% errores	E cuad
Weka	Aleatoria	Minkowski 1	False	88,67	11,33	159,3 *
Weka	Aleatoria	Minkowski 1	True	89,33	10,67	47,779 *
Weka	Aleatoria	Minkowski 2	False	88,67	11,33	78,94
Weka	Aleatoria	Minkowski 2	True	89,33	10,67	7
MDP6	Aleatoria	Minkowski 1	False	88,67	11,33	223,21
MDP6	Aleatoria	Minkowski 2	False	88,67	11,33	78,95
MDP6	Aleatoria	Minkowski 7'5	False	88,67	11,33	50,06
MDP6	Aleatoria	Chebyshev	False	88,67	11,33	82,2
MDP6	Pert_aleatoria	Minkowski 1	False	88,67	11,33	223,21
MDP6	Pert_aleatoria	Minkowski 2	False	89,33	10,67	78,95
MDP6	Pert_aleatoria	Minkowski 7'5	False	88,67	11,33	50,06
MDP6	Pert_aleatoria	Chebyshev	False	66,67	33,33	48,36
MDP6	Cent_aleatorios	Minkowski 1	False	88,67	11,33	223,21
MDP6	Cent_aleatorios	Minkowski 2	False	66,67	33,33	78,94
MDP6	Cent_aleatorios	Minkowski 7'5	False	88,67	11,33	50,08
MDP6	Cent_aleatorios	Chebyshev	False	51,33	48,67	87,51
MDP6	Div_espacio	Minkowski 1	False	88,67	11,33	223,21
MDP6	Div_espacio	Minkowski 2	False	66,67	33,33	145,28
MDP6	Div_espacio	Minkowski 7'5	False	66,67	33,33	91,2
MDP6	Div_espacio	Chebyshev	False	66,67	33,33	87,51

*Sum of within clusters distances.

Figura 4.4: Resultados de clasificación para el fichero iris.arff

4.4. Variabilidad de los resultados

Este apartado se dedica única y exclusivamente a la observación de la variabilidad de los resultados de una ejecución a otra, con los mismos parámetros, de nuestro software.

Las pruebas se han realizado con el uso de la distancia Euclídea, variando el parámetro K en el conjunto {3, 5, 8} y la inicialización entre las 4 posibilidades disponibles.

Para cada una de las combinaciones de parámetros tratadas, se ha ejecutado 10000 veces, obteniéndose en cada una el valor del error cuadrático medio.

El proceso se ha automatizado y ejecutado mediante el ejecutable *MDP6_var_res.jar*, desarrollado especialmente para la tarea.

Posteriormente, los 10000 valores de cada ejecución se han tratado con el software R para obtener las medidas estadísticas deseadas.

Los ficheros resultado de este experimento se encuentran en */resultados/variabilidad_resultados*, siguiendo la misma estructuración que los del apartado 4.2. A continuación se muestran los resultados obtenidos.

Los resultados obtenidos por los algoritmos en los que intervienen variables de carácter aleatorio arrojan resultados distintos en cada repetición mientras que el algoritmo de inicialización por división del espacio resulta siempre en un resultado idéntico siempre que tratemos con el mismo fichero de datos.

La distribución de los resultados se muestra gráficamente mediante franjas en los valores en los que tienen tendencia a converger los resultados de la aplicación de los algoritmos. Estas franjas

	K=	***3***	***5***	***8***
Inicialización aleatoria	Mínimo	3794	1274	362,8
	1er Cuadrante	3815	1902	539,8
	Mediana	4093	2019	1452,2
	Media	4349	2123	1112,8
	3er Cuadrante	4093	2154	1504
	Máximo	7039	5775	5678,4
	Varianza	925665,7	478596,4	250481,7
Pertenencia aleatoria	Mínimo	7039	1274	378,4
	1er Cuadrante	3794	1827	876,5
	Mediana	3815	1964	1519,1
	Media	4093	1908	1292,3
	3er Cuadrante	3997	2024	1658,4
	Máximo	4093	3395	2441,9
	Varianza	57161,97	1274	235878,9
Centroides aleatorios	Mínimo	3794	115044,2	378,4
	1er Cuadrante	3815	1827	876,5
	Mediana	4093	1964	1513,4
	Media	3992	1914	1277,4
	3er Cuadrante	4093	2024	1653,7
	Máximo	6655	3483	2441,9
	Varianza	50895,6	1339	236035,2
División espacial	Mínimo	4093	118689,5	1129
	1er Cuadrante	4093	1339	1129
	Mediana	4093	1339	1129
	Media	4093	1339	1129
	3er Cuadrante	4093	1339	1129
	Máximo	4093	1339	1129
	Varianza	0	0	0

Figura 4.5: Medidas estadísticas para las diferentes pruebas

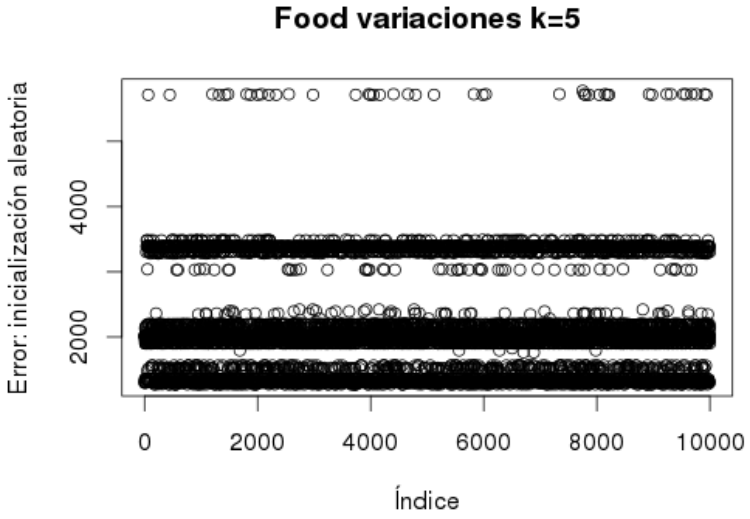


Figura 4.6: Franjas de convergencia

son similares entre el algoritmo de pertenencia aleatoria y el de centroides aleatorios.

Si comparamos los índices estadísticos, observamos que los algoritmos que mejor puntúan en base a su media son los de pertenencia y centroides aleatorios para aquellas pruebas realizadas sobre 3 y 5 clusters, sin embargo, la puntuación de la inicialización aleatoria es mejor para 8 clusters.

La división del espacio otorga siempre el mismo resultado teniendo por tanto varianza cero. Este algoritmo no ha puntuado como último en ninguna de las pruebas por lo que podría ser un candidato a cota realista del desempeño posible. Por otra parte, en el caso de k=3 encontramos curioso el hecho de que todas las medianas sean iguales y coincida exactamente con el resultado arrojado por la división del espacio. Esto se puede deber a que el algoritmo converja en un valor de 4093 para una gran cantidad de condiciones iniciales distintas siendo esta cifra ,por tanto , un atractor clásico.

El algoritmo que mayor varianza arroja en todos los experimentos ha sido el de inicalización aleatoria siendo la varianza de los otros dos algoritmos evaluables en función de este parámetro similar entre si. Pese a tener la tendencia a arrojar un mejor resultado con un número grande de clusters hay que tener en cuenta esta gran variabilidad de resultados para el algoritmo de inicialización aleatoria.

También parece claro que un mayor número de clusters favorece una disminución de la varianza para el algoritmo de inicialización aleatoria. Esto tiene sentido si entendemos que a mayor número de clusters la distancia de las distintas instancias al centroide del cluster es menor en comparativa y dado que seleccionamos una instancia como centroide, es más probable que exista una convergencia rápida.

Sin embargo, para los algoritmos de pertenencia aleatoria y centroides aleatorios parece que sucede al contrario esto puede deberse al hecho de que las pruebas se han realizado sobre un número limitado de iteraciones siendo estos dos algoritmos de convergencia más lenta.

4.5. Análisis crítico y discusión de resultados

Los resultados de la aplicación de este programa sobre los datos de prueba han sido variados y complejos en muchos casos. La gran variedad de parámetros de entrada establece una gran forma de realizar múltiples pruebas de precisión consiguiendo en muchos casos unos resultados que resultan óptimos siempre y cuando seamos capaces de adaptar los parámetros del programa a las necesidades del conjunto de instancias estudiadas.

Los diferentes algoritmos de inicialización ofrecen distintas características a tener en cuenta. Para algunos algoritmos encontramos que funcionan mejor comparativamente para aquellas situaciones en las que menos clusters haya, tal es así en el caso de los centroides aleatorios y la pertenencia aleatoria.

Sin embargo, se da la circunstancia de que para una mayor cantidad de clusters el algoritmo de inicialización que parece adaptarse mejor es el clásico de inicialización aleatoria.

Este no es el único parámetro a tener en cuenta ya que también conviene estudiar la varianza de las distintas inicializaciones para considerar si lo que se busca es una solución precisa a la vez que estable. La cantidad de atributos influye también en la selección del método de inicialización ya que ciertos algoritmos tienen una mayor precisión comparativa cuando se aumenta el número de dimensiones[1].

Por otro lado, el software desarrollado comparado con el programa **Weka**, con el que trabajamos en clase, arroja resultados similares, en ciertas ocasiones mejores y en otras peores debido sobre todo a que con la variedad de posibilidades que implementa nuestro software somos capaces de adaptarnos mejor a las situaciones que plantean los distintos espacios en estudio.

Por último, cabe señalar la cantidad de resultados reseñables que hemos sido capaces de alcanzar gracias al estudio en masa de los resultados proporcionados, estos resultados no siempre están reflejados en la práctica por ser resultados parciales e intermedios.

4.6. Rendimiento del software

El software desarrollado posee una complejidad computacional que difiere en función del algoritmo de inicialización. El algoritmo de clustering que usamos, por si mismo, posee un orden computacional kn siendo k el número de iteraciones que realiza el software y n el número de instancias en evaluación.

Para las inicializaciones de centroide aleatorio, división del espacio e inicialización aleatoria, el software no necesita recorrer las instancias que se encuentran en evaluación, sin embargo, para el de pertenencia aleatoria es necesario acceder a todas estas instancias.

Esta necesidad de acceso no varía el orden de magnitud aunque aplicado al algoritmo tenemos que el orden resultante es de $k(1+n)$ interpretando las variables de la misma forma que el enunciado anterior.

Dicho esto, el algoritmo en cualquiera de sus vertientes y sobre los archivos de prueba que hemos usado, aún en el caso de realización sistemática de experimentos, no ha presentado ningún inconveniente temporal. Este rendimiento podría ser evaluado de nuevo sobre archivos con una mayor cantidad de información para conocer realmente en qué cotas es inviable su aplicación y debería ser reestructurado siguiendo criterios de rendimiento.

[1] Hecho que aún no hemos estudiado en profundidad

Capítulo 5

Conclusiones

5.1. Motivación para la realización de *Clustering*

Explorar un conjunto de instancias con el objetivo de crear agrupaciones (clusters) donde todas tienen algo en común, alguna característica similar.

Hay diferentes estrategias a considerar. Si, por ejemplo, queremos clasificar datos, digamos 1000 instancias, en tres clases diferentes sin tener que examinarlas una por una, podemos hacer clustering k-medias con k = 3, y obtendríamos directamente los tres grupos. Como contrapartida, puede que muchas instancias hayan quedado mal clasificadas en la clase que no le corresponde. Este es el enfoque que hemos llevado en el desarrollo de nuestro software.

En este mismo ejemplo, otra opción es crear más clusters, digamos 10. Posteriormente, podríamos examinar una instancia de cada cluster (o dos o tres para tener más certeza) de forma que a todas las instancias de ese cluster les asignamos la clase resultante de esa instancia. De esta forma se reducen los errores al etiquetar las 1000 instancias, pero en vez de examinarlas todas, únicamente se deben examinar 10, 20 o 30 de las 1000 instancias originales.

5.2. Conclusiones de los resultados

El software desarrollado es **válido**, cuanto menos, realiza el mismo trabajo que Weka, **mejorándolo** en algunos casos, si bien las instancias de trabajo están **limitadas** a aquellas definidas por atributos **numéricos**.

A destacar el hecho de que, en el experimento de clasificación supervisada, las diferentes combinaciones de parámetros obtengan la misma tasa de aciertos. Si bien es un hecho que no sabemos explicar muy bien, también es cierto que en la prueba nos hemos limitado al fichero *iris* sin probar otras opciones.

5.3. Conclusiones generales

El tiempo invertido en el proyecto ha sido mucho mayor conforme se acercaba la fecha de entrega original. Debemos mejorar en organización, especialmente al principio de la práctica, donde

un diseño poco especificado con ciertas cosas *.en el aire"* nos llevaron a diseñar al implementar.

También el hecho de probar y perfeccionar el funcionamiento de **todo** el código antes de comenzar con la extracción de resultados experimentales nos limitó en el análisis de los mismos y en la retroalimentación del algoritmo implementado.

Resumiendo, hay que mejorar la planificación, dividir el proyecto en partes para poder mejorarlo incrementalmente.

Pese a los problemas, la solución implementada ofrece una variedad bastante amplia de parámetros posibles a utilizar, teniendo 4 tipos de inicialización, dos métricas diferentes - en el caso de la métrica Minkowski ofrece otro parámetro más que variar- y la posibilidad de controlar el número de iteraciones tanto por la especificación de un máximo de vueltas, como por la especificación del margen de variación que se debe alcanzar hasta dar por válidos los centroides identificados.

La semana extra de trabajo nos ha servido para exponer resultados útiles que ayuden a determinar la calidad de nuestro software, resultados para los cuales se ha observado la gran utilidad que tiene la automatización de la obtención de las mismas para ser capaces de procesar un gran volumen de datos obteniendo los resumen concluyentes.

5.4. Propuestas de mejora

La entrada a nuestro programa es un factor un tanto restrictivo en el sentido de que el único tipo de fichero soportado es el **CSV**. Inicialmente, la intención era implementar también la carga de ficheros **ARFF**, que no ha sido posible hacerla por falta de planificación principalmente.

En cuanto a inicializaciones, hemos implementado 4 tipos diferentes. De los tres, el correspondiente a la división de espacio es el que más puede mejorar, ya que para su implementación no hemos llegado a consultar bibliografía extra. El mapeo de aleatorios sería otra mejora que garantizaría estabilidad a tres de las inicializaciones.

Una restricción que podríamos eliminar está asociada al tipo de datos tratados, el algoritmo solo es capaz de lidiar con atributos de tipo numérico, siendo imposible realizar la carga de instancias que contengan valores no numéricos. De hecho, algunos de los ficheros han tenido que ser preprocesados para poder trabajar con ellos.

También teníamos un problema con la representación de los resutados, ya que los valores reales se presentaban con todos los decimales que tuvieran. Esto fue solucionado en la semana extra, haciendo que al imprimir los valores, se redondeasen previamente a 3 o 5 decimales.

Se podrían mejorar los programas auxiliares creados, de forma que permitan configurar más parámetros (las Ks a usar por ejemplo) y permitan experimentar a los usuarios de la aplicación.

Por último, cabe mejorar el apartado de resultados del algoritmo utilizando métodos concretos de la evaluación de clusters. Por ejemplo, la implementación del índice silueta, que sería un indicador mejor que el error cuadrático y nos permitiría saber directamente lo bien o mal que nuestro algoritmo realiza el clustering.

Bibliografía

[1] Alicia Pérez. Minería de datos, tema 9: Clasificación no-supervisada (clustering).

Capítulo 6

Valoración subjetiva

Alcance de objetivos

Utilidad de la tarea

Dificultad

Tiempo de trabajo

Sugerencias de mejora

Críticas