Comparación de diferentes algoritmos de clustering en la estimación de coste en el desarrollo de software

Miguel Garre[1] Juán José Cuadrado[1] Miguel Ángel Sicilia[1]

[1]Dept. de Ciencias de la Computación

ETS Ingeniería Informática

Universidad de Alcalá

Ctra. Barcelona km 33.6 - 28871

Alcalá de Henares, Madrid

miguel.garre,jjcg,msicilia,@uah.es

Resumen

En el presente trabajo se aplican diferentes algoritmos de clustering para la estimación de costo software. Se trata de tres tipos diferentes de algoritmos de clustering: COBWEB (clustering jerárquico), k-medias (particionado), y EM (probabilístico). Cada uno de ellos se aplica sobre los proyectos de la base de datos ISBSG, realizando una segmentación de los mismos en diferentes grupos de afinidad. Esta medida de afinidad es diferente, según el algoritmo de clustering de que se trate, obteniéndose por tanto diferentes segmentos según el algoritmo que se utilice. Las medidas que se utilizarán para comparar la bondad de los segmentos obtenidos por cada uno de los algoritmos de clustering son MMRE y PRED(.3), ámpliamente utilizadas en la literatura sobre el tema. De los tres algoritmos utilizados destaca EM como el que mejor segmentación realiza, a continuación k-medias y por último COBWEB. La razón estriba en que la forma que tiene COBWEB de obtener los segmentos no es la más adecuada para los datos suministrados al mismo, EM y k-medias ofrecen mejores resultados ya que se trata de métodos de la misma familia, y adecuados para la naturaleza de los datos.

1. Introducción

El llevar a cabo un proyecto software, lleva asociado un esfuerzo de dedicación en cuanto a tiempo y dinero se refiere. El poder estimar, preferiblemente, en las primeras etapas de construcción de software, estos valores, ayuda en gran medida en la consecución del proyecto. Muchos han sido, desde sus inicios en los años 40, los proyectos software que han fracasado por una mala planificación en el tiempo y coste. Es por ello por lo que desde estos primeros momentos surgieron multitud de métodos de estimación de coste software, para ayudar en el desarrollo de los mismos.

Tantos métodos de estimación han sido agrupados en diferentes clasificacíones, la primera se debe a Barry Boehm [6], en la que clasifica a los métodos de estimación de coste software en: modelos algorítmicos, juicio de expertos, analogía, parkinson, price to win, top-down y bottom-up. A esta clasificación le sigue la de DeMarco [7], la de Conte [8]: modelos históricos-experimentales, modelos estadísticos, modelos teóricos, y modelos compuestos. Más tarde, en 1991 Kitchenham [9], clasifica en opinión de expertos, analogía, descomposición, y ecuaciones de estimación, los diferentes métodos de estimación. Fairley [10] en 1992 ofrece esta otra clasificación, modelos empíricos, basados en regresión, y basados en la teoría. En 1997, Walkerden y Jeffery [11] nos ofrecen otra

clasificación, modelos empíricos paramétricos, modelos empíricos no-paramétricos, modelos analógicos, modelos teóricos y modelos heúrísticos. Recientemente, Barry Boehm [12] ofrece otra clasificación, actualización de la realizada en 1981, en la cual aparecen nuevas clasificaciones, en total tenemos: técnicas basadas en modelos, técnicas basadas en expertos, orientados al aprendizaje, modelos dinámicos, modelos basados en técnicas de regresión, y modelos compuestos-bayesianos. Dentro de los métodos orientadas al aprendizaje, Boehm menciona las redes neuronales y el estudio de casos (originados a partir de los modelos por analogía). Por último, en el año 2001, Wieczorek [13] clasifica los métodos de estimáción en, métodos basados en modelos y métodos no-basados en modelos, dentro de los primeros tenemos genéricos (propietarios y no propietarios) y específicos (conducidos por los datos y compuestos).

Si nos centramos en los métodos orientados al aprendizaje, según la clasificación de 2001 de Boehm, actualmente además de las redes neuronales y el razonamiento basado en casos (CBR) (Shepperd [16]), se están aplicando otras técnicas de machine learning para intentar mejorar el proceso de estimación, sobre los proyectos actuales, que revisten una gran complejidad. Tales técnicas son las de programación genética, sistemas fuzzy, inducción de reglas, CART (Classification and Regression Trees) y combinaciones de éstas. Ejemplos de ello, pueden ser los artículos de Ali Idri, en los que aplica sistemas difusos combinados con analogías [15, 14], o los de Dolado en los que utiliza programación genética [17, 18] para estimar el costo software. En el caso de inducción de reglas, C. Mair compara esta técnica con redes neuronales y razonamiento basado en casos en [19]. L. Briand, ha realizado diferentes análisis y comparativas, en las que interviene CART, [21, 20]. Y por último mencionar el artículo de A. Lee [22], en el que se utilizan técnicas de clustering para el entrenamiento de una red neuronal en la estimación de coste software.

Este trabajo es continuación de una serie de ellos en los que se aplican técnicas de clustering para segmentar un conjunto de proyectos software, y sobre éstos aplicar modelos matemáticos de estimación, cosa que hasta ahora no se había experimentado. Estos artículos son los de J. Cuadrado et al. [4, 5] y los de M. Garre et al. [1, 2, 3]. En estos trabajos se obtienen una serie de grupos de proyectos, a partir de la base de proyectos ISBSG versión 8 (Internacional Software Benchmarking Standard Group[1]), y sobre cada uno de ellos se aplica un modelo matemático multiplicativo de la forma $e = as^b$, donde e es el esfuerzo estimado y s alguna medida del tamaño del proyecto. Las constantes a y b se obtienen mediante análisis de regresión sobre los proyectos de cada grupo. Los resultados obtenidos (respecto a MMRE y PRED(.3)), como se muestra en los trabajos anteriormente mencionados, mejoran los que se consiguen al utilizar una única ecuación para todos los proyectos.

Tras comprobar la bondad de estas técnicas en el proceso de estimación de costo software, es necesario realizar una comparativa de diferentes algoritmos de clustering, con la intención de ver cúal o cúales de ellos se comportan de forma más efectiva para realizar dicha estimación. Se compararán tres de estos algoritmos, COBWEB, k-medias y EM (utilizado en [4, 5, 1, 2, 3]), ofreciendo los resultados obtenidos. El resto del artículo se estructura de la siguiente forma: en la sección 2 se realizará una definición de la metodología de clustering, se mostrará una clasificación de los diferentes algoritmos de clustering, así como una descripción de los algoritmos COBWEB, EM y k-medas. En la sección 3 se aplican estos algoritmos a la base de datos ISBSG y se comparan los resultados ofrecidos por los mismos. Finalmente en la sección 4 aparecen las conclusiones obtenidas a partir de la comparativa realizada.

2. Clustering

El proceso de *Clustering* consiste en la división de los datos en grupos de objetos similares. Para medir la similaridad entre objetos se suelen

[1] http://www.isbsg.org/

utilizar diferentes formas de distancia: distancia euclídea, de Manhatan, de Mahalanobis, etc. El representar los datos por una serie de clusters, conlleva la pérdida de detalles, pero consigue la simplificación de los mismos. Clustering es una técnica más de Machine Learning, en la que el aprendizaje realizado es no-supervisado (*unsupervised learning*). Desde un punto de vista práctico, el clustering juega un papel muy importante en aplicaciones de data mining, tales como exploración de datos científicos, recuperación de la información y minería de texto, aplicaciones sobre bases de datos espaciales (tales como GIS o datos procedentes de astronomía), aplicaciones Web , marketing, diagnóstico médico, análisis de ADN en biología computacional, y muchas otras. En el presente trabajo lo utilizaremos como complemento a los modelos matemáticos, en la estimación de coste software.

Una gran variedad de algoritmos de clustering han surgido en los últimos años, los cuales se puden clasificar en:

- Métodos Jerárquicos
  - Algoritmos Aglomerativos
  - Algoritmos Divisivos
- Métodos de Particionado y Recolocación
  - Clustering Probabilístico
  - Métodos de los k-vecinos (*k-medoids*)
  - Métodos de las k-medias
  - Algoritmos Basados en Densidad
    - Clustering de Conectividad Basada en Densidad *Density-Based Connectivity Clustering*
    - Clustering basado en Funciones de Densidad
- Métodos Basados en Rejillas
- Métodos Basados en la Co-Ocurrencia de Datos Categóricos
- Clustering Basado en Restricciones
- Algoritmos para Datos de Grandes Dimensiones
  - Clustering Subespacial
  - Técnicas de Co-Clustering

A continuación veremos más en detalle los que serán utilizados en este trabajo.

2.1. COBWEB

Se trata de un algoritmo de *clustering jerárquico*. COBWEB [23], se caracteriza porque utiliza aprendizaje incremental, esto es, realiza las agrupaciones instancia a instancia. Durante la ejecución del algoritmo se forma un árbol (*árbol de clasificación*) donde las hojas representan los segmentos y el nodo raíz engloba por completo el conjunto de datos de entrada. Al principio, el árbol consiste en un único nodo raíz. Las instancias se van añadiendo una a una y el árbol se va actualizando en cada paso. La actualización consiste en encontrar el mejor sitio donde incluir la nueva instancia, operación que puede necesitar de la reestructuración de todo el árbol (incluyendo la generación de un nuevo nodo *anfitrión* para la instancia y/o la fusión/partición de nodos existentes) o simplemente la inclusión de la instancia en un nodo que ya existía. La clave para saber cómo y dónde se debe actualizar el árbol la proporciona una medida denominada *utilidad de categoría*, que mide la calidad general de una partición de instancias en un segmento. La reestructuración que mayor utilidad de categoría proporcione es la que se adopta en ese paso. El algoritmo es muy sensible a otros dos parámetros:

Acuity Este parámetro es muy necesario, ya que la utilidad de categoría se basa en una estimación de la media y la desviación estándar del valor de los atributos, pero cuando se estima la desviación estándar del valor de un atributo para un nodo en particular, el resultado es cero si dicho nodo sólo contiene una instancia. Así pues, el parámetro acuity representa la medida de error de un nodo con una sola instancia, es decir, establece la varianza mínima de un atributo.

Cut-off Este valor se utiliza para evitar el crecimiento desmesurado del número de

segmentos. Indica el grado de mejoría que se debe producir en la utilidad de categoría para que la instancia sea tenida en cuenta de manera individual. En otras palabras: cuando no es suficiente el incremento de la utilidad de categoría en el momento en el que se añade un nuevo nodo, ese nodo se corta, conteniendo la instancia otro nodo ya existente.

Además COBWEB pertenece a los métodos de aprendizaje conceptual o basados en modelos. Esto significa que cada cluster se considera como un modelo que puede describirse intrinsicamente, más que un ente formado por una colección de puntos.

Al algoritmo COBWEB no hay que proporcionarle el número exacto de clusters que queremos, sino que en base a los parámetros anteriormente mencionados encuentra el número óptimo. La implementación utilizada en el presente trabajo, de este algoritmo es la de Weka[2].

2.2. EM

EM pertenece a una familia de modelos que se conocen como *Finite Mixture Models*, los cuales se pueden utilizar para segmentar conjuntos de datos. Dentro de la clasificación de la sección 2, encajaría dentro de los Métodos de Particionado y Recolocación, en concreto Clustering Probabilístico.

Se trata de obtener la FDP (Función de Densidad de Probabilidad) desconocida a la que pertenecen el conjunto completo de datos. Esta FDP se puede aproximar mediante una combinación lineal de NC componentes, definidas a falta de una serie de parámetros $\{\Theta\} = \cup\{\Theta_j \forall j = 1..NC\}$, que son los que hay que averiguar,

$$P(x) = \sum_{j=1}^{NC} \pi_j p(x;\Theta_j) \text{ con } \sum_{j=1}^{NC} \pi_j = 1 \quad (1)$$

donde π_j son las probabilidades *a priori* de cada cluster cuya suma debe ser 1, que también forman parte de la solución buscada, $P(x)$ denota la FDP arbitraria y $p(x;\Theta_j)$ la función de densidad del componente j. Cada cluster se corresponde con las respectivas muestras de datos que pertenecen a cada una de las densidades que se mezclan. Se pueden estimar FDP de formas arbitrarias, utilizándose FDP normales n-dimensionales, t-Student, Bernoulli, Poisson, y log-normales. Aquí se modelarán los datos mediante distribuciones normales, por ser éstas las más comunes.

El ajuste de los parámetros del modelo requiere alguna medida de su bondad, es decir, cómo de bien encajan los datos sobre la distribución que los representa. Este valor de bondad se conoce como el *likelihood* de los datos. Se trataría entonces de estimar los parámetros buscados Θ, maximizando este likelihood (este criterio se conoce como *ML-Maximun Likelihood*). Normalmente, lo que se calcula es el logaritmo de este likelihood, conocido como *log-likelihood* ya que es más fácil de calcular de forma analítica. La solución obtenida es la misma, gracias a la propiedad de monotonicidad del logaritmo. La forma de esta función log-likelihood es:

$$L(\Theta,\pi) = log \prod_{n=1}^{NI} P(x_n) \quad (2)$$

donde NI es el número de instancias, que suponemos independientes entre si.

El algoritmo *EM*, procede en dos pasos que se repiten de forma iterativa:

Expectation Utiliza los valores de los parámetros, iniciales o proporcionados por el paso *Maximization* de la iteración anterior, obteniendo diferentes formas de la FDP buscada.

Maximization Obtiene nuevos valores de los parámetros a partir de los datos proporcionados por el paso anterior.

después de una serie de iteraciones, el algoritmo EM tiende a un máximo local de la función L.

Finalmente se obtendrá un conjunto de clusters que agrupan el conjunto de proyectos original. Cada uno de estos cluster estará definido por los parámetros de una distribución

[2] http://www.cs.waikato.ac.nz/∼ml/weka/

normal.

La implementación utilizada no es la de Weka, ya que la implementación que lleva a cabo del algoritmo EM lleva asociada la premisa de la independencia de los atributos utilizados, nada más lejos de la realidad existente entre el esfuerzo y los puntos de función. Por lo tanto se ha desarrollado una implementación de EM en lenguaje C que se adapta mucho mejor a las características del problema de estimación de coste software. Para una mayor descripción de la misma se puede consultar [3].

2.3. K-medias

Se trata de un algoritmo clasificado como Método de Particionado y Recolocación. El método de las k-medias [24, 25], es hasta ahora el más utilizado en aplicaciones científicas e industriales. El nombre le viene porque representa cada uno de los clusters por la media (o media ponderada) de sus puntos, es decir, por su centroide. Este método únicamente se puede aplicar a atributos numéricos, y los *outliers* le pueden afectar muy negativamente. Sin embargo, la representación mediante centroides tiene la ventaja de que tiene un significado gráfico y estadístico inmediato. La suma de las discrepancias entre un punto y su centroide, expresado a través de la distancia apropiada, se usa como función objetivo.
La función objetivo, suma de los cuadrados de los errores entre los puntos y sus centroides respectivos, es igual a la varianza total dentro del propio cluster. La suma de los cuadrados de los errores se puede razionalizar, como el negativo del log-likelihood, para modelos mixtos que utilicen distribuciones normales. Por lo tanto, el método de las k-medias se puede derivar a partir del marco probabilístico (ver subsección *Clustering Probabilístico* del libro de Mitchell [26]).

Existen dos versiones del método de las k-medias. La primera es parecida al algoritmo EM, y se basa en dos pasos iterativos: primero reasigna todos los puntos a sus centroides más cercanos, y en segundo lugar recalcula los centroides de los nuevos grupos creados en el anterior. El proceso continua hasta alcanzar un criterio de parada (por ejemplo que no se realicen nuevas reasignaciones). Esta versión se conoce como algoritmo de Forgy [27].
La segunda versión [28] reasigna los puntos basándose en un análisis más detallado de los efectos causados sobre la función objetivo al mover un punto de su cluster a otro nuevo. Si el traslado es positivo, se realiza, en caso contrario se queda como está.

A diferencia de los anteriores algoritmos, COBWEB y EM, k-medias necesita la previa especificación del número de clusters que se desean obtener.La implementación utilizada en el presente trabajo, de este algoritmo es también la ofrecida por Weka.

3. Caso de Estudio

Para realizar el estudio de los diferentes algoritmos de clustering, primero es preciso realizar un filtrado de los proyectos de la base de datos ISBSG, que será utilizada. Esto se hará en la sección 3.1. Una vez preparados los datos, se aplican los algoritmos COBWEB, EM y k-medias, sobre los mismos, obteniéndose unos resultdos que se ofrecen en la sección 3.2. La comparación de estos resultados se verá en la sección 3.3.

3.1. Preparación de los datos

Se utilizó la base de datos de proyectos ISBSG-8, la cual contiene información sobre 2028 proyectos. Esta base de datos contiene información sobre tamaño, esfuerzo, y otras características de un proyecto. El primer paso de limpieza consistió en eliminar de la base de datos todos los proyectos con valores numéricos no válidos o nulos en los campos esfuerzo ('Summary Work Effort' en ISBSG-8) y tamaño ('Function Points' en ISBSG-8). Además todos los proyectos cuyo valor para el atributo 'Recording Method' fuese distinto de *Staff Hours* también fueron eliminados. La razón es que se considera que el resto de formas de considerar el esfuerzo son subjetivas. Por ejemplo *Productive Time* es una magnitud difícil de valorar en un contexto organizativo.

Otro aspecto a tener en cuenta para la limpieza de los datos es la forma en la que se obtuvieron los diferentes valores de los puntos de función. En concreto se examinó el valor del atributo 'Derived count approach', descartando todos los proyectos que no hubiesen utilizado como forma de estimar los puntos de función métodos como IFPUG, NESMA, Albretch o Dreger. Las diferencias entre los métodos IFPUG y NESMA tienen un impacto despreciable sobre los resultados de los valores de los puntos de función [29]. Las mediciones basadas en las técnicas Albretch no se eliminaron ya que, de hecho IFPUG es una revisión de estas técnicas. De la misma forma el método Dreger [30] es simplemente una guía sobre las mediciones IFPUG. Por último se procede a la eliminación de los proyectos con valores nulos para el atributo tiempo ('Project Elapsed Time'). Finalmente el estudio se realiza sobre una base de datos de 1569 proyectos.

3.2. Aplicación de los algoritmos de clustering

En esta sección veremos los resultados que se obtienen al aplicar los tres algoritmos de clustering bajo estudio, sobre la base de datos ISBSG, una vez preparados los datos, como hemos visto en la sección 3.1.

3.2.1. COBWEB

Este algoritmo admite dos parámetros, como hemos visto en la sección 2.1, *cutoff* y *acuity*. Los valores que Weka propone para éstos es de *acuity*=1.0 y *cutoff*=0.0028209479177387815. Con estos valores no se obtiene ningún cluster, por lo que es necesario modificarlos. Según el significado del *cutoff* parece lógico el decrementar este valor, para obtener un mayor número de clusters, para ello se le asigna un valor de *cutoff*=0.00028209479177387815. Obteniéndose los siguientes resultados:

- Número de fusiones: 386.
- Número de divisiones: 896.
- Número de clusters: 1792.

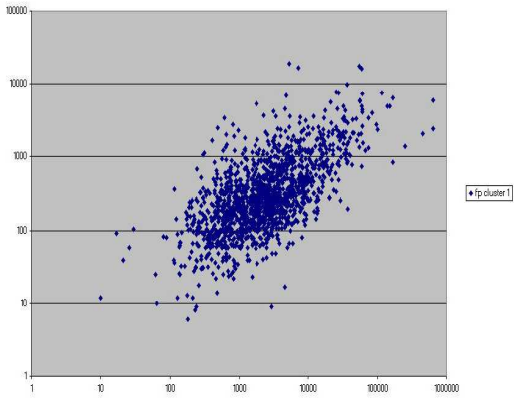

Figura 1: Clusters obtenidos por el algoritmo COBWEB.

De los 1792 clusters, 1593 están formados por un elemento, y los restantes por ninguno. El número tan elevado de clusters hace pensar que el valor elegido no es el más adecuado, es necesario incrementarlo. Tras varios intentos se utiliza un valor de *cutoff*=0.0018750338970421, el cual ofrece unos resultados aceptables:

- Número de fusiones: 283.
- Número de divisiones: 222.
- Número de clusters: 95.

Se hicieron nuevas intentonas, pero ninguna dio un número de clusters menor. Todos los resultados coincidian en lo siguiente: se obtienen 94 clusters que cubren 114 proyectos repartidos uniformemente, mientras que el cluster restante engloba a 1479 proyectos.

En la figura 1 se puede ver el cluster más importante. La escala de esta figura, así como las siguientes, es logarítmica, para poder apreciar mejor la forma de los segmentos obtenidos.

Los valores de MMRE y PRED(.3) son los que se muestran en el cuadro 1. Éstos valores resultan de utilizar la recta de regresión del cluster obtenido, cuyos coeficientes a y b también se ofrecen en dicho cuadro.

	Cluster 1
N° Proyectos	1479
a	26.52
b	0.7899
MMRE	133.56 %
PRED(.3)	23.56 %

Cuadro 1: Valores MMRE y PRED(.3) para COBWEB.

3.2.2. EM

Este algoritmo proporciona una segmentación de los proyectos en 9 clusters. La representación gráfica se puede ver en la figura 2.

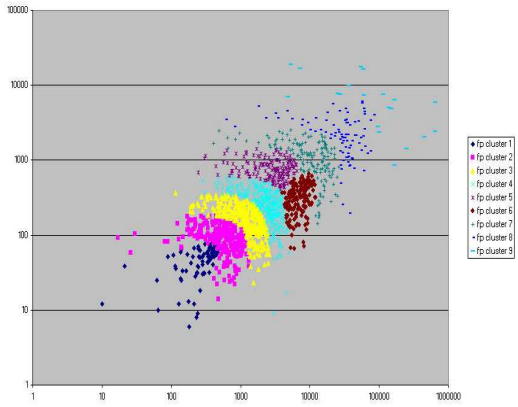

Figura 2: Clusters obtenidos por el algoritmo EM.

Los valores de los parámetros se ofrecen a continuación en los cuadros 2, 3 y 4. Los valores de MMRE y PRED(.3) han resultado utilizando las respectivas rectas de regresión de cada cluster, cuyos coeficientes a y b también se ofrecen en dichos cuadros.

Para el cluster 9, formado únicamente por 20 proyectos no se realiza ningún análisis por consider que los resultados ofrecidos no serían fidedignos, por el pequeño número de elementos de este segmento.

	CLUS 1	CLUS 2	CLUS 3
N° Proyectos	74	249	311
Probabilidad	0.046	0.1569	0.2655
Media e	284.68	616.15	0.179
Media fp	48.24	95.95	175.26
Desv Stand e	128.96	309.55	594.94
Desv Stand fp	20.46	39.27	76.67
Coef Cor e-fp	0.5931	-0.2218	-05015
a	36.6	2829	34640
b	0.5012	-0.3794	-0.675
MMRE	64.88 %	74.23 %	41.54 %
PRED(.3)	56.75 %	46.37 %	53.69 %

Cuadro 2: Resultados del algoritmo EM. Clusters 1, 2 y 3.

	CLUS 4	CLUS 5	CLUS 6
N° Proyectos	343	161	170
Probabilidad	0.2022	0.1198	0.1173
Media e	2367.9	3072	6679.83
Media fp	278.8	714.9	317.82
Desv Stand e	1063.23	1590.45	2490.04
Desv Stand fp	128.86	302.13	141.02
Coef Cor e-fp	-0.5213	-0.1	0.3625
a	41730	44890	2500
b	-0.5351	-0.4322	0.1781
MMRE	39.46 %	71.47 %	22.14 %
PRED(.3)	59.18 %	41.61 %	69.41 %

Cuadro 3: Resultados del algoritmo EM. Clusters 4, 5 y 6.

3.2.3. K-medias

Al algoritmo de las k-medias hay que proporcionarle de antemano el número de clusters en los que queremos que se segmente la base de proyectos. Este dato se obtiene a partir del algoritmo EM, ya que éste obtiene este dato de forma óptima (vease [3]). Por lo tanto el algoritmo de k-medias se aplica sobre la base de datos de proyectos ISBSG para un número de clusters de 9 y un valor para la semilla de 10. La representación gráfica de los clusters obtenidos se puede ver en la figura 3.

En este caso, el número de elementos por cluster es el siguiente: cluster 1, 4 proyectos;

	CLUS 7	CLUS 8	CLUS 9
Nº Proyectos	155	87	20
Probabilidad	0.1027	0.0621	0.0133
Media e	10984.97	31144.76	161796.42
Media fp	1123.3	2356.22	7509.93
Desv Stand e	6226.94	18749.39	194440.38
Desv Stand fp	540.69	1383.26	5647
Coef Cor e-fp	-0.3855	0.0069	-0.4755
a	672200	581400	-
b	-0.6172	-04131	-
MMRE	80.42 %	139.69 %	-
PRED(.3)	40 %	41.37 %	-

Cuadro 4: Resultados del algoritmo EM. Clusters 7, 8 y 9.

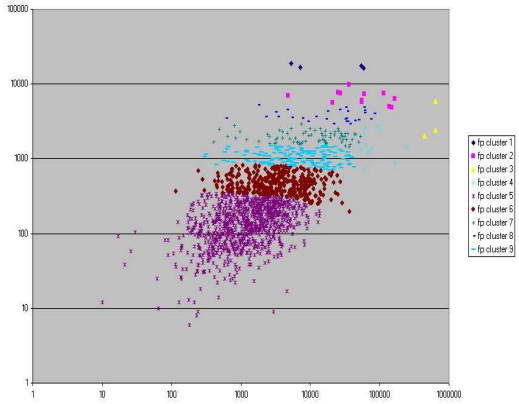

Figura 3: Clusters obtenidos por el algoritmo de k-medias.

cluster 2, 12 proyectos; cluster 3, 3 proyectos; cluster 4, 11 proyectos; cluster 5, 920 proyectos; cluster 6, 361 proyectos; cluster 7, 74 proyectos; cluster 8, 28; y cluster 9, 156. Por lo tanto, solamente se analizarán los clusters 5, 6, 7, 8 y 9, los restantes se descartan por tener un número de proyectos muy pequeño. Al igual que en el caso anterior, podemos ver los resultados de aplicar este algoritmo, tras 50 iteraciones, en los cuadros 5 y 6. Igualmente, los valores de MMRE y PRED(.3) han resultado utilizando las respectivas rectas de regresión de cada cluster.

	CLUS 5	CLUS 6	CLUS 7
Nº Proyectos	920	361	74
Probabilidad	0.59	0.23	0.05
Media e	1811.23	5257.41	15869.70
Media fp	151.18	479.22	1980.13
Desv Stand e	1846.71	5066.92	13546.90
Desv Stand fp	81.10	137.29	324.03
a	7652	77390	730200
b	0.5505	-0.5078	-0.5733
MMRE	109.68 %	116.81 %	193.5 %
PRED(.3)	24.34 %	23.26 %	13.51 %

Cuadro 5: Resultados del algoritmo k-medias. Clusters 5, 6 y 7.

	CLUS 8	CLUS 9
Nº Proyectos	28	156
Probabilidad	0.02	0.1
Media e	28417.57	10664.88
Media fp	3794.85	1069.98
Desv Stand e	23373.81	9871.42
Desv Stand fp	647.19	212.73
a	3699000	686300
b	-0.6579	-0.6708
MMRE	229.58 %	155.92 %
PRED(.3)	10.71 %	14.74 %

Cuadro 6: Resultados del algoritmo k-medias. Clusters 8 y 9.

3.3. Comparación de los resultados obtenidos

Si se comparan los valores de MMRE y PRED(.3) de los clusters obtenidos por los algoritmos bajo estudio, se observa que el método EM destaca sobre los otros dos.

COBWEB ofrece un único cluster significativo, con resultados no muy diferentes de los que se obtendrían sin realizar segmentación alguna, considerándo todos los proyectos para una única curva de regresión. De hecho el cluster obtenido es una representación de toda la base de datos, eliminando algunos outliers.

EM y k-medias ofrecen segmentos de *similares* formas, aunque han agrupado los proyectos de diferente foma, ver figuras 2 y 3. Ello se debe al hecho de que ámbos forman parte de una misma familia de algoritmos de clustering y tienen bases comunes. Sin embargo EM destaca claramente sobre k-medias, ofreciendo mucho mejores valores de MMRE y PRED(.3).

4. Conclusiones

El algoritmo COBWEB, perteneciente a la familia de clustering jerárquico, no parece adecuado para la segmentación de proyectos software, ya que tiende a agrupar a la mayoría en un solo segmento, cosa que no es deseable de cara a la mejora en la estimación de costo software.

EM y k-medias, perteneciendo a la familia de algoritmos de particionado y recolocación, ofrecen mejores resultados que COBWEB, indicando que para tareas de segmentación de proyectos software son más adecuados que éste. Comparando EM y k-medias, se observa que los segmentos son diferentes, k-medias agrupa en un sólo cluster los mismos proyectos que EM divide en varios. Es decir, EM realiza una división más específica que k-medias.

Se puede concluir que el algoritmo EM, siendo un algoritmo que realiza clustering probabilístico, es más adecuado que el algoritmo k-medias, para segmentar una base de datos de proyectos software, con el fin de mejorar la estimación del costo software.

Para próximos trabajos se profundizará en el estudio del algoritmo EM, utilizando otras formas de distribución de datos, así como la utilización de varios atributos con dependencia e independencia entre ellos. Así mismo se realizarán comparaciones entre EM con algoritmos de clustering de otras familias, para comprobar la bondad de los mismos.

Referencias

[1] Garre, M., Cuadrado, J.J. and Sicilia, M.A., *Recursive segmentation of software projects for the estimation of development effort*, en Proceedings of the ADIS 2004 Workshop on Decision Support in Software Engineering, CEUR Workshop proceedings Vol. 120, disponible en http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-120, 2004.

[2] Garre, M., Charro, M., *Estimación del esfuerzo de un proyecto software utilizando el criterio MDL-EM y componentes normales N-Dimensionales. Aplicación a un caso práctico*, Revista de Procesos y Métricas de las Tecnologías de la Información (RPM)ISSN: 1698-2029, VOL. 2, Nº 1, Marzo 2005, 13-24. Asociación Española de Sistemas de Informáticos (AEMES), 2005.

[3] Garre M., Cuadrado J.J., Sicilia, M.A., Charro M, Rodríguez D., *Segmented Parametric Software Estimation Models: Using the EM algorithm with the ISBSG 8 database*, Information Technology Interfaces, Croacia 20-23 junio 2005.

[4] J. Cuadrado Gallego, Daniel Rodríguez, Miguel Ángel Sicilia. *Modelos Segmentados de estimación del esfuerzo de desarrollo del software: un caso de estudio con la base de datos ISBSG*. Revista de Procesos y Métricas de las Tecnologías de la Información (RPM). VOL. 1, Nº 2, Agosto 2004, 25-30 ISSN: 1698-2029.

[5] Cuadrado J.J., Sicilia, M.A., Garre M., Rodríguez D., *An empirical study of process-related attributes in segmented software cost-estimation relationships*, Journal of Systems and Software, 2005.

[6] Boehm, B., *Software Engineering Economics*, Vol. 10, Prentice-Hall, 1981.

[7] DeMarco, T., *Controlling Software Projects*, Yourdan Press, 1982.

[8] Conte, S.D., Dunsmore, H. E. & Shen, V.Y., *Software Engineering Metrics and Models*, Benjamin/Cumming Co., Inc., Menlo Park, 1986.

[9] Fenton, N.E., *Software metrics: a rigorous approach*, Chapman & Hall, Londres, 1991.

[10] Fairley, R.E., *Recent advances in software estimation techniques*, en International Conference on Software Engineering, ACM New York, USA, 1992.

[11] Walkerden, F. & Jeffery, D., *Software cost estimation: A review of models, process, and practice*, Advances in Computers 44, 59-125, 1997.

[12] Boehm, B., Abts, C. & Chulani, S., *Software development cost estimation approaches - a survey*, Annals of Software Engineering 10, 177-205, 2000.

[13] Wieczorek, I. & Briand, L., *Resource estimation in software engineering*, Technical Report, International Software Engineering Research Network, 2001.

[14] Idri, A. and Abran, A., *Fuzzy Case-Based Reasoning Models for Software Cost Estimation*,2002.

[15] Idri, A. and Abran, A. and Khoshgoftaar, T. M., *Fuzzy Analogy: A new Approach for Software Cost Estimation*, Proceedings of the 11th International Workshop on Software Measurements, 93-101, Montreal, Canada, 2001.

[16] Shepperd, M., and Schofield, C., *Estimating software project effort using analogies*, IEEE Transactions on Software Engineering, 1997.

[17] Dolado, J.J., *On the problem of the software cost function*, Information and Software Technology, VOL. 43, 61-72, 2001.

[18] Dolado, J. J. and Fernández L., *Genetic Programming, Neural Networks and Linear Regression in Software Project Estimation*, INSPIRE III, Process Improvement thorough Training and Education, 155-171, The British Computer Society, 1998.

[19] Mair, C. and Kadoda, G. and Lefley, M. and Keith, P. and Schofield, C. and Shepperd, M. and Webster, S., *An investigation of machine learning based prediction systems*, The Journal of Systems and Software, VOL. 53, 23-29, 2000.

[20] Briand, L. and Langley, T. and Wieczorek, I., *Using the European Space Agency Data Set: A Replicated Assessment and Comparison of Common Software Cost Modeling*, Proceedings of the 22th International Conference on Software Engineering, 377-386, Limerik, Ireland, 2000.

[21] Briand, L. C. and El Emam, K. and Maxwell, K. and Surmann, D. and Wieczorek, I., *An Assessment and Comparison of Common Cost Software Project Estimation Methods*, Proc. International Conference on Software Engineering, ICSE 99, 313-322, 1999.

[22] Lee, A. and Cheng, C. H. and Balakrishann, J., *Software development cost estimation: Integrating neural network with cluster analysis*, Information & Management, N°34, 1-9, 1998.

[23] Fisher, D., *Knowledge acquisition via incremental conceptual clustering*, Machine Learning, VOL. 2, 139-172, 1987.

[24] Hartigan, J., *Clustering Algorithms*, John Wiley & Sons, New York, 1975.

[25] Hartigan, J. and Wong, M., *Algorithm AS139: A k-means clustering algorithm*, Applied Statistics, VOL. 28, 100-108, 1979.

[26] Tom M. Mitchell. *Machine Learning*. McGraw-Hill. (1997).

[27] Forgy, E., *Cluster analysis of multivariate data: Efficiency versus interpretability of classification*, Biometrics, VOL. 21, 768-780, 1965.

[28] Duda, R. and Hart, P., *Pattern Classification and Scene Analysis*, John Wiley & Sons, 1973.

[29] NESMA (1996). *NESMA FPA Counting Practices Manual* (CPM 2.0).

[30] Dreger, J. Brian. *Function Point Analysis*. Englewood Cliffs, NJ: Prentice Hall, 1989.