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

Miguel Garre, Juan José Cuadrado, Miguel A. Sicilia, Daniel Rodríguez, Ricardo Rejas
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,daniel.rodriguezg}@uah.es

Resumen

Los modelos de estimación de coste software que obtienen una única relación matemática entre el esfuerzo y algún otro atributo característico de los proyectos software, proporcionan buenos resultados cuando la base de datos de proyectos, a partir de la que mediante métodos de regresión se obtiene la relación anteriormente mencionada, está formada por proyectos homogéneos. Sin embargo, para bases de datos de proyectos procedentes de muy diversas fuentes, tales como la base de proyectos de ISBSG formada por miles de proyectos heterogéneos, el utilizar una única relación matemática para representar a todos estos proyectos, no ofrece tan buenos resultados como si los proyectos fuesen homogéneos. En este trabajo se plantea, como mejora del proceso de estimación, segmentar la base de datos ISBSG en diferentes grupos de proyectos mediante la utilización de tres algoritmos de agrupamiento diferentes: COBWEB, EM, y k-means, de manera que para cada uno de estos grupos (formados por proyectos homogéneos entre sí) se obtenga una relación matemática diferente. La segmentación llevada a cabo por estos algoritmos mejora la estimación con respecto al modelo que utiliza la base de datos sin segmentar. Por otra parte si se comparan entre sí los resultados obtenidos al aplicar cada uno de ellos, se observa que el algoritmo que presenta un mejor comportamiento es EM debido a su naturaleza probabilista.

Abstract

The software cost estimation models that obtain an only mathematical relation between the effort and some other attribute characteristic of the software projects, provide good results when the projects data base, from which by means of regression methods the mentioned relation is obtained previously, is formed by homogenous projects. Nevertheless, for projects data bases coming from very diverse sources, such as the ISBSG projects data base formed by thousands of heterogeneous projects, using an only mathematical relation to represent all these projects, does not offer so good results as if the projects were homogenous. In this work one considers, like improvement of the estimation process, to segment the ISBSG data base in different groups from projects by means of the use of three different clustering algorithms: COBWEB, EM, and k-means, so that for each one of these clusters (formed by homogenous projects to each other) a different mathematical relation is obtained. The carried out segmentation by these algorithms improves the estimation with respect to the model that uses the data base without segmenting. On the other hand if the

results obtained when applying are compared to each other, it is observed that the algorithm whom a better behaviour presents is EM due to its probabilistic nature.

Palabras clave: algoritmo EM, estimación de coste software, clustering, segmentación, ISBSG.

1. Introducción

El desarrollo de un proyecto software, tiene asociado un esfuerzo de dedicación en cuanto a tiempo y dinero se refiere. El poder estimar estos valores, preferiblemente, en las primeras etapas de construcción de software (cuando esta información es más valiosa para el desarrollo del proyecto), ayuda en gran medida en la consecución del mismo. Muchos han sido los proyectos software que han fracasado por una mala planificación en el tiempo y coste. Es por ello por lo que existen multitud de métodos de estimación de coste software, para ayudar en el desarrollo de los mismos.

La multitud de métodos de estimación existentes han sido agrupados en diferentes clasificaciones, 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, precio para ganar[1], de arriba hacia abajo[2] y de abajo hacia arriba[3]. 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], los 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] ofrecen otra clasificación, modelos empíricos paramétricos, modelos empíricos noparamétricos, modelos basados en analogías, modelos teóricos y modelos heurísticos. Recientemente, Boehm [12] ofrece otra clasificación, actualización de la realizada en 1981, en la cual aparecen nuevas clasificaciones, las agrupa en: 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 orientados al aprendizaje, Boehm menciona las redes

[1] *Price to win.*
[2] *Top-down.*
[3] *Bottom-up.*

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 estimación 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 se centra la atención 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 aprendizaje automático[4] para intentar mejorar el proceso de estimación. Tales técnicas son las de programación genética, sistemas difusos, inducción de reglas, CART (Árboles de Regresión y Clasificación[5]) 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 coste software. En el caso de inducción de reglas, C. Mair compara esta técnica con redes neuronales y razonamiento basado en casos [19]. L. Briand, ha realizado diferentes análisis y comparativas, en las que interviene CART, [21, 20]. Por último mencionar el artículo de A. Lee [22], en el que se utilizan técnicas de *clustering*[6] para el entrenamiento de una red neuronal en la estimación de coste software.

El presente artículo es continuación de una serie de trabajos publicados anteriormente en los que se aplican técnicas de *clustering* para segmentar un conjunto de proyectos software sobre los que se aplican modelos matemáticos de estimación. 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*[7]), 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 (Magnitud media del error relativo[8]) y PRED(0,3) (Nivel de

[4] *Machine learning*.
[5] *Classification and Regresión Trees.*
[6] Agrupamiento
[7] http://www.isbsg.org
[8] *Mean Magnitude of Relative Error*.

predicción[9]), mejoran los que se consiguen al utilizar una única ecuación para todos los proyectos. La medida MMRE indica el valor medio de los errores cometidos al realizar la estimación, en este caso del esfuerzo, para cada uno de los proyectos disponibles. Por otro lado PRED(*l*), indicaría el porcentaje de estimaciones que cometen un error menor a *l*. Cuanto menor sea MMRE y mayor sea PRED(*l*) mejor será el modelo obtenido, Conte [8] indica como valores aceptables para MMRE $\leq$ 0,25, y PRED(0,3) $\geq$ 0,75.

En los trabajos anteriormente citados [4, 5, 1, 2, 3] se ha utilizado EM como algoritmo para realizar el proceso de *clustering,* demostrando su bondad. En este trabajo se van a utilizar otros algoritmos diferentes a EM, con el fin de comparar los resultados que se obtienen con cada uno de ellos en el proceso final de estimación. Se compararán tres algoritmos, COBWEB, k-means[10] y EM, ofreciéndose los resultados que se obtienen al aplicar cada uno de ellos en el proceso de agrupamiento. 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-medias. 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 similitud entre objetos se suelen 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 Aprendizaje Automático, en la que el aprendizaje realizado es no supervisado. Desde un punto de vista práctico, el *clustering* juega un papel muy importante en aplicaciones de minería de datos, 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,

[9] *Prediction Level*
[10] K-medias

marketing, diagnóstico médico, análisis de ADN en biología computacional, y muchas otras. En el presente trabajo se utilizará como complemento a los modelos matemáticos en la estimación de coste software.

En los últimos años han surgido una gran variedad de algoritmos de *clustering*, a continuación se mostrarán 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 intrínsicamente, 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[11].

2.2 EM

EM pertenece a una familia de modelos que se conocen como *Finite Mixture Models*[12], los cuales se pueden utilizar para segmentar conjuntos de datos. Es un método de 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,

$$P(x) = \sum_{j=1}^{NC} \pi_j \; p(x;\theta_j), \qquad \sum_{j=1}^{NC} \pi_j = 1. \qquad (1)$$

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

[11] http://www.cs.waikato.ac.nz/_ml/weka/
[12] Modelos de mezcla finitos

bondad se conoce como el *likelihood*[13] 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:

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

donde NI es el número de instancias, que se suponen 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 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, cosa que en el caso del esfuerzo y de los puntos de función no se cumple. 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

[13] Verosimilitud

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*[14] 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 racionalizar, 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 continúa hasta alcanzar un criterio de parada (por ejemplo que no se realizan 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 resultados que se ofrecen en la sección 3.2. La comparación de estos resultados se verá en la sección 3.3.

[14] Valores desacordes con el resto

3.1 Preparación de los datos

Se utilizó la base de datos de proyectos ISBSG-8, la cual contiene información sobre 2028 proyectos heterogéneos, procedentes de diferentes organizaciones. Esta base de datos contiene información sobre tamaño, esfuerzo, y otras características de un proyecto, considerándose veintiún de estos atributos (los más importantes y representativos). 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 (sin ajustar) 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, tras el proceso de limpieza, el estudio se realiza sobre una base de datos de 1569 proyectos.

Para obtener más información sobre los atributos de la base de datos ISBSG se puede visitar su página web (www.isbsg.org) y buscar la descripción de los mismos.

3.2 Aplicación de los algoritmos de clustering

En esta sección se mostrarán 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 se ha visto en la sección 3.1.

3.2.1 COBWEB

Este algoritmo admite dos parámetros, como se ha visto en la sección 2.1, *cutoff* y *acuity*. Los valores que Weka propone para éstos es de *acuity*=1.0 y *cutoff*=0.0028. Con estos valores no se obtiene ningún cluster, por lo que es necesario modificarlos. Según el significado del *cutoff* parece lógico decrementar este valor, para obtener un mayor número de clusters, para ello se le asigna un valor de *cutoff*=0.00028, obteniéndose 1292 clusters.

Cada cluster está formado por unos pocos elementos, de manera que se tienen muchísimos clusters de muy pocos proyectos. El número tan elevado de clusters hace evidente que el valor elegido no es el más adecuado, es necesario incrementarlo. Tras varios intentos se utiliza un valor de *cutoff*=0.0018, obteniéndose 74 clusters.

Se hicieron nuevas pruebas, pero ninguna ofreció un número de clusters menor. De los clusters obtenidos, 73 cubren 90 proyectos repartidos uniformemente, mientras que el cluster restante engloba a 1479 proyectos.

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

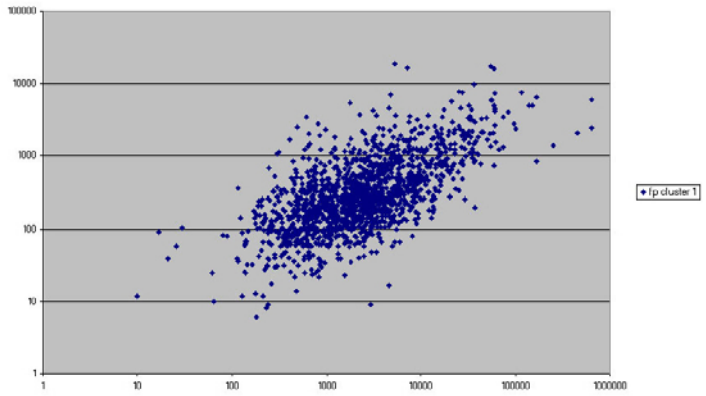

Figura 1: Cluster principal obtenido por el algoritmo COBWEB.

Los valores de MMRE y PRED(0,3) son los que se muestran en la tabla 1. Estos valores resultan de utilizar la recta de regresión del cluster obtenido, cuyos coeficientes *a* y *b* también se ofrecen en dicha tabla.

	Cluster 1
Nº Proyectos	1479
a	26,52
b	0,7899
MMRE	133,56%
PRED(0,3)	23,56%

Tabla 1. Valores MMRE y PRED(0,3) 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.

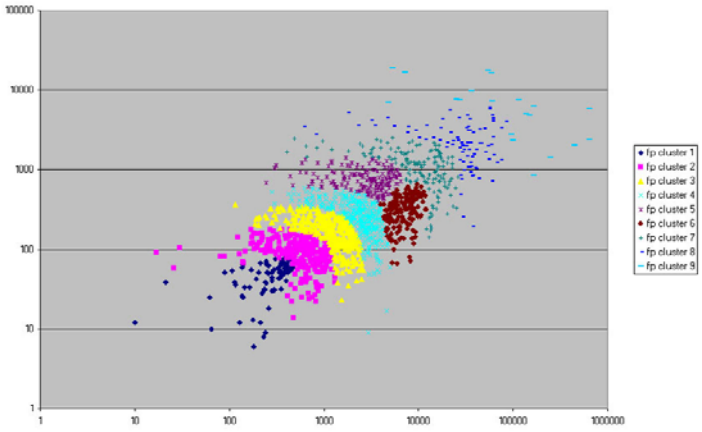

Figura 2: Clusters obtenidos por el algoritmo EM.

Los valores de los parámetros se ofrecen a continuación en la tabla 2. Los valores de MMRE y PRED(0,3) han resultado utilizando las respectivas rectas de regresión de cada cluster, cuyos coeficientes *a* y *b* también se ofrecen en dicha tabla.

Nº Cluster	1	2	3	4	5	6	7	8	9
Nº Proyectos	74	249	311	343	161	170	155	87	20
Probabilidad	0,046	0,1569	0,2655	0,2022	0,1198	0,1173	0,1027	0,0621	0,0133
Media e	284,68	616,15	0,179	2367,9	3072	6679,83	10984,97	31144,76	161796,42
Media fp	48,24	95,95	175,26	278,8	714,9	317,82	1123,3	2356,22	7509,93
Desv Stand e	128,96	309,55	594,94	1063,23	1590,45	2490,04	6226,94	18749,39	194440,38

Desv Stand fp	20,46	39,27	76,67	128,86	302,13	141,02	540,69	1383,26	5647
a	36,6	2829	34640	41730	44890	2500	672200	581400	-
b	0,5012	-0,3794	-0,675	-0,5351	-0,4322	0,1781	-0,6172	-0,4131	-
MMRE	64,88%	74,23%	41,54%	39,46%	71,47%	22,14%	80,42%	139,69%	-
PRED(0,3)	56,75%	46,37%	53,69%	59,18%	41,61%	69,41%	40%	41,37%	-

Tabla 2. Resultados del algoritmo EM.

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

3.2.3 K-medias

Al algoritmo de las k-medias hay que proporcionarle de antemano el número de clusters en los que se quiere segmentar la base de proyectos. Este dato se obtiene a partir del algoritmo EM, ya que éste obtiene este dato de forma óptima (véase [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.

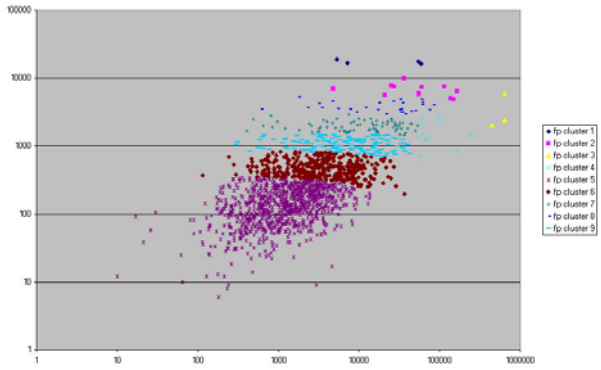

Figura 3: Clusters obtenidos por el algoritmo k-medias.

En este caso, el número de elementos por cluster es el siguiente: cluster 1, 4 proyectos; 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, se pueden ver los resultados de aplicar este algoritmo, tras 50 iteraciones, en la tabla 3. Los valores de MMRE y PRED(0,3) resultan de aplicar rectas de regresión de cada cluster.

	Cluster 5	Cluster 6	Cluster 7	Cluster 8	Cluster 9
Nº Proyectos	920	361	74	28	156
Probabilidad	0,59	0,23	0,05	0,02	0,1
Media e	1811,23	5257,41	15869,70	28417,57	10664,88
Media fp	151,18	479,22	1980,13	3794,85	1069,98
Desv Stand e	1846,71	5066,92	13546,90	23373,81	9871,42
Desv Stand fp	81,10	137,29	324,03	647,19	212,73
a	7652	77390	730200	3699000	686300
b	0,5505	-0,5078	-0,5733	-0,6579	-0,6708
MMRE	109,68%	116,81%	193,5%	229,58%	155,92%
PRED(0,3)	24,34%	23,26%	13,51%	10,71%	14,74%

Tabla 3. Resultados del algoritmo k-medias.

3.3 Comparación de los resultados obtenidos

Si se comparan los valores de MMRE y PRED(0,3) 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, considerando 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*.

Este algoritmo no es 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 coste software.

EM y k-medias ofrecen segmentos de similares formas, aunque han agrupado los proyectos de diferente forma, ver figuras 2 y 3. Ello se debe al hecho de que ambos 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(0,3).

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.

4. Conclusiones

A continuación se mostrarán las conclusiones que se obtienen al aplicar cada uno de los algoritmos de *clustering* utilizados en este trabajo.

Este trabajo ha mostrado diferentes algoritmos de *clustering* que se han utilizado en la estimación de coste software. Para compararlos entre sí se han utilizado dos medidas ampliamente utilizadas y conocidas en la estimación, MMRE y PRED(0,3). Teniendo en cuenta estos datos, se observa que el algoritmo EM es el mejor algoritmo de los tres que se puede utilizar para la estimación de coste software, según el proceso descrito.

En general, siempre que se desee realizar una estimación de coste para un nuevo proyecto software, y se disponga de una base de datos heterogénea de proyectos históricos para ello, la mejor opción consistirá en utilizar el algoritmo EM tal y como se ha mostrado a lo largo del artículo. Los valores estimados utilizando EM son más parecidos a los reales, que si se utiliza COBWEB o k-medias para tal fin.

Se puede concluir que el algoritmo EM, siendo un algoritmo que realiza *clustering* probabilístico, es más adecuado que el algoritmo k-medias y COBWEB, para segmentar una base de datos de proyectos software, con el fin de mejorar la estimación del coste 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. y Sicilia, M.A., "Recursive segmentation of software projects for the estimation of development effort", *Proceedings of the ADIS 2004 Workshop on Decision Support in Software Engineering*, CEUR Workshop proceedings, 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) Vol. 2, nº 1, Marzo 2005, 2005, pp. 13-24.

[3] Garre M., Cuadrado J.J., Sicilia, M.A., Charro M. y 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, pp. 25-30.

[5] Cuadrado-Gallego, J.J., Sicilia, M.A., Rodríguez, D. y Garre M., "An empirical study of process-related attributes in segmented software cost-estimation relationships," *Journal of Systems and Software*, Vol. 3, nº 79, marzo 2006, pp. 351-361.

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

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

[8] Conte, S.D., Dunsmore, H. E. y Shen, V.Y., "Software Engineering Metrics and Models", Benjamin/Cummings, 1986.

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

[10] Fairley, R.E., "Recent advances in software estimation techniques", *Proceedings of the 14th international conference on Software engineering*, Melbourne, Australia, 1992, pp. 382 – 391.

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

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

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

[14] Idri, A., Abran, A. y Khoshgoftaar, T. M., *Fuzzy Case-Based Reasoning Models for Software Cost Estimation*, Springer-Verlag, 2004.

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

[16] Shepperd, M. y Schofield, C., "Estimating software project effort using analogies", *IEEE Transactions on Software Engineering*, Vol. 23, nº 11, 1997, pp. 736-743.

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

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

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

[20] Briand, L., Langley, T. y 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, Limerick, Ireland, 2000.

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

[22] Lee, A., Cheng, C. H. y Balakrishann, J., "Software development cost estimation: Integrating neural network with cluster analysis", *Information & Management*, nº 34, 1998, pp. 1-9.

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

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

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

[26] Mitchell, T. M., *Machine Learning*. McGraw-Hill, 1997.

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

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

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

[30] Dreger, J. Brian. *Function Point Analysis*. Prentice Hall, 1989.