STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LIV**, Number 1, 2009

OVERVIEW OF FUZZY METHODS FOR EFFORT ESTIMATION BY ANALOGY

MILITON FRENŢIU AND HORIA F. POP

Abstract. Estimation of cost and time to develop a software product P is a well known problem. Among the possible ways to do it is analogy with old finished projects. We describe an experiment using different versions of the Fuzzy c-Means algorithm, to find projects similar to P. Then the information about these projects is used to estimate the effort needed to build P.

1. Introduction

Estimation of cost and optimal time to develop a software product is a well known problem. There are various methods to solve this problem:

- expert opinion, i.e. guessing on personal experience;
- analogy with already finished projects;
- decomposition of the system into its smallest components and estimating the effort to build each component;
- models using mathematical formulas to compute the cost.

There are some models for cost estimation. Here we mention COCOMO I [Boe81], Albrecht's Function Points [Alb83], and COCOMO II [Boe85]. The COCOMO I is based on the size of the product, which is unknown at the beginning of the software process. Albrecht's model is based on the requirements and uses the information about the input data, the needed results, the files used in the system and the inquiries made by the user. COCOMO II effort estimation equations are adjusted basic equation using 17 cost drivers and five scale factors. These cost drivers are personnel attributes, process attributes, product attributes, and computer attributes.

Received by the editors: April 5, 2009.

2000 *Mathematics Subject Classification.* 68N30.

1998 *CR Categories and Descriptors.* D.2.3 [**Software**] : Software Engineering – *Coding Tools and Techniques*; I.5.1 [**Computing Methodologies**] : Pattern Recognition – *Models* – *Fuzzy set*; G.3 [**Mathematics of Computing**] : Probability and Statistics – *Data analysis*.

Key words and phrases. Software metrics, Cost estimation, Fuzzy clustering.

To predict by analogy the cost (effort) and time to develop a software system P means to find the most similar finished system S and to use its known characteristics for prediction [She96]. The most similar system is that one which is closest to P in the Euclidean space.

First problem that arise is to define and compute the similarity of two projects S and P. We need some important characteristics of these projects, and the possibility to obtain these characteristics from the requirements of the projects. They appear in Albrecht's model, or in COCOMO2, and may be quantifications for the complexity of the system, the required reliability, the number of inputs, the number of outputs, the number of files, the number of screens and so forth. If we use n characteristics to describe a project, it is represented as a point in the n-dimensional Euclidean space. The similarity of two projects may be defined as the distance between the corresponding points.

Estimating the cost of P by analogy is reduced to finding the closest previously completed similar project S. Then the actual data from the project S is extrapolated to estimate the cost of P.

An improvement to the above described method is to select some projects most similar to P and to deduce the cost for the proposed project from data of these most similar completed projects. Here we suggest using both the well-established Fuzzy Hierarchical Clustering procedure and as well a restricted approach thereof, to find the class C of the most similar projects to P.

The cost of P is computed as a weighted average of $Cost(S)$ for $S \in C$, i.e.

$$Cost(P) = \sum w(S) \times Cost(S), S \in C$$

where the costs $Cost(S)$ are estimations of $Cost(P)$ based on information available for project S, the weights $w(S)$ are determined based on the membership degrees of all items in the class and the sum of all $w(S)$ for all $S \in C$ is 1. Alternate estimations of the cost will be studied in a future paper.

2. Restricted Fuzzy Clustering

Let us consider a set of classified objects, $X = \{x^1, \ldots, x^p\} \in \mathbb{R}^s$ and the fuzzy partition $P = \{A_1, \ldots, A_n\}$ corresponding to the cluster substructure of the set X. Let $x^0 \in \mathbb{R}^d$ be an object that needs to be classified with respect to the fuzzy partition P.

The algorithm we are presenting here computes the optimal fuzzy partition $\tilde{P}$ corresponding to the set $\tilde{X} = X \cup \{x^0\}$, by using a mechanism similar to Fuzzy n-means, with the difference that the membership degrees of the objects in X to the classes A_i, $i = 1, \ldots, n$ may not be modified.

In what follows we consider a metric d in the Euclidean space $\mathbb{R}^s$. We will suppose that d is norm induced, so $d(x,y) = (x-y)^T M(x-y), x, y \in \mathbb{R}^s$, where M is a symmetrical and positively defined matrix.

The objective function we have in mind for our problem is similar to that for the Fuzzy n-Means Algorithm:

$$\tilde{J}(\tilde{P}, L) = \sum_{i=1}^{n} \sum_{j=0}^{p} \left(A_i(x^j)\right)^2 d^2\left(x^j, L^i\right),$$

with the mention that $A_i(x^j)$ are kept constant for each i and for $j = 1, \ldots, p$.

The main result with respect to determining the fuzzy partition $\tilde{P}$ and its representation L minimizing the function $\tilde{J}$ is the following

Theorem. (i) The fuzzy partition $\tilde{P} = \{A_1, \ldots, A_n\}$ has the minimum value of the function $J(\cdot, L)$ if and only if

$$A_i(x^0) = \frac{1}{\sum_{k=1}^{n} \frac{d^2(x^0, L^i)}{d^2(x^0, L^k)}}. \tag{1}$$

(ii) The set of prototypes $L = \{L^, \ldots, L^n\}$ has the minimum value of the function $J(\tilde{P}, \cdot)$ if and only if

$$L^i = \frac{\sum_{j=0}^{p} \left(A_i(x^j)\right)^2 x^j}{\sum_{j=0}^{p} \left(A_i(x^j)\right)^2}. \tag{2}$$

With this result, the optimal membership degrees for x^0 to the classes A_i will be determined using an iterative method in which $\tilde{J}$ is successively minimized with respect to $\tilde{P}$ and L. The process will start with the initialization of prototypes L^i to the values that correspond to the fuzzy membership degrees of the original fuzzy partition P.

The resulted algorithm, **Restrictive Fuzzy n-Means Clustering Algorithm**, follows:

S1 Let us have X and P as given variables.
S2 Determine the initial positions of the prototypes L^i according to value of P.
S3 Determine the membership degrees $A_i(x^0)$, $i = 1, \ldots, n$, using relation (1).
S4 Determine the new positions of prototypes L^i, $i = 1, \ldots, n$, using relation (2).

S5 If the new positions of the prototypes L^i are close enough to the former positions, then **stop**, else return to step S3.

3. Hierarchic Fuzzy Clustering with incomplete data

The problem of incomplete data is extremely important [Hat01]. If a cluster substructure of a data set with incomplete data is rewquired, the option of ignoring the incomplete data items altogether is not realistic, because that would assume ignoring useful, available data. Hathaway and Bezdek have proposed in 2001 a few stragegies to cope with the incomplete data problem [Hat01]. Their approach has been to extend the Fuzzy c-means algorithm in such a way as to accept incomplete data as well. We have used the approach of Hathaway and Bezdek and produced hierarchic clustering versions thereof.

The main issue is that, in computing the fuzzy membership degrees, the square distances $d^2(x^j, L^i)$ from the data item x^j to the class prototype L^i cannot be computed for incomplete data. Assuming that we use the Euclidean metric, we would need to compute

$$D_{ij} = \sum_{k=1}^{s} \left(x_k^j - v_k^i \right)^2 . \tag{3}$$

However, for some values of j and k, x_k^j may be missing. In our paper we are going to use the *Partial Distance Strategy* of Hathaway and Bezdek [Hat01]. Namely, we are going to compute this sum considering only the available values and we are going to scale the result to take into account the missing dimensions. As such, we are actually going to compute

$$\tilde{D}_{ij} = \frac{s}{\sum_{k=1}^{s} I_{jk}} \sum_{k=1}^{s} I_{jk} \left(x_k^j - v_k^i \right)^2 , \tag{4}$$

where $I_{jk} = 1$ if the value x_k^j is available, and $I_{jk} = 0$ if the value is unavailable, in which case the difference will not be computed.

Similarily, the new prototypes will have to be computed by taking into account only the available data:

$$L_k^i = \frac{\sum_{j=1}^{p} I_{jk} \left(A_i(x^j)\right)^2 x_k^j}{\sum_{j=0}^{p} I_{jk} \left(A_i(x^j)\right)^2}, \tag{5}$$

where I_{jk} have the same meaning as above.

The resulted algorithm, **Hierarchic Fuzzy Clustering with Incomplete Data**, follows:

S1 Let us consider the data set X and the initial fuzzy partition P.
S2 Determine the positions of prototypes L^i, $i = 1, \ldots, n$, using relation (5).
S3 Determine the membership degrees $A_i(x^0)$, $i = 1, \ldots, n$, using dissimilarities from (4).
S4 If the new fuzzy partition is close enough to the former fuzzy partition, then **stop**, else return to step S2.

4. Non-binary Divisive Hierarhic Fuzzy Clustering

One of the key problems of divisive clustering is the choice for binary split at every level of the clustering hierarchy. This is a very simple, intuitive and effective approach. While it leads to desired results in most of the cases, there are situations where the data does not show a natural binary split-up. Here a different approach must be used.

Our approach is to generalize the binary divisive approach introduced in [Dum88].

The central point of the Fuzzy Hierarchic Divisive method is the binary polarization index. Thus, considering a binary fuzzy partition $P = \{C_1, C_2\}$, of the fuzzy set C, the partition separation index is defined as

$$R(P) = \frac{\sum_{i=1}^{2} \sum_{k=1}^{n} C_{i,1/2}(x_k)}{\sum_{k=1}^{n} C(x_k)}$$

where

$$C_{i,t}(x) = \begin{cases} C_i(x) & C_i(x) > t \\ 0 & \text{otherwise} \end{cases}$$

Now, instead of considering a binary partition, let us consider one with p arity, $P = \{C_1, \ldots, C_p\}$. We define the generalized polarization index quite naturally as

$$R(P) = \frac{\sum_{i=1}^{p}\sum_{k=1}^{n} C_{i,1/2}(x_k)}{\sum_{k=1}^{n} C(x_k)}.$$

Our key problem is then to obtain the best partition arity at every node of the classification tree. Fortunately, this is a very simple issue from a constructive point of view. Our assumption has been that the data does not display a binary structure. So, at every node of the classification tree we are going to construct both a 2-partition and a 3-partiton. The decision of which is best to adopt is based on the higher partition polarisation index. By working recursively, in this way, we are going to obtain a classification tree where each node has either two or three children, as appropriate.

5. Experiment

We have used the information we had about 19 student software projects to estimate the cost of a new project (numbered 20). This information refers to the complexity, reliability, considered difficulty (all three metrics have values between 1 and 5, denoting very low, low, normal, high and very high), number of inputs, number of outputs, number of files and number of screens). Also, the computed function points and the cost of these projects are known.

The final partition produced using this 20×8 data matrix contains three classes. All the three classes are well separated. The core elements of the classes have quite high fuzzy membership degrees, while only a few of elements per class, not members of the class, have fuzzy membership degrees reasonably large.

We are estimating the cost of project 20 using the projects member of the very same class project 20 is a member of. Very similar results are obtained using the other suggested approach, i.e. Restricted Fuzzy Clustering, where we first find the cluster substructure of projects 1-19 and then embed the 20-th project as an extra data item in the former fuzzy partition.

6. Conclusions and Future Research

We have used fuzzy classification as a way to find the projects similar to our project P. There are usually more than one very similar projects, and these are the projects found in the same class with P. Then, to predict the cost/effort needed to built P, we have used the information about all these projects similar to P.

It is considered that estimating by analogy is a superior technique to estimation via algorithmic model in at least some circumstances [She96]. However, it requires to maintain a database with information about the finished systems. Oftenly, the information about finished projects is incomplete, and we cannot find and add the missing information about these old projects. It would be useful to obtain the classification in the absence of some information. Factorial analysis [Wat67] may also be used to predict the effort to build a new project from the information about finished projects.

7. Acknowledgement

This material is based upon work supported by the Romanian National University Research Council under award PN-II no. ID_550/2007.

References

[Alb83] Albrecht A. J., Gaffney Jr J. E. (1983). Software function, source lines of code, and development effort prediction: A software science validation. IEEE Transactions on Software Engineering 9 (6), 639-648.

[Boe81] Boehm B. W. (1981). Software engineering economics, Prentice-Hall: Englewood Cliffs, N.J.

[Boe85] Boehm B. W., Clark B., et al. (1995). Cost models for future software life cycle processes: COCOMO 2.0. Annals of Software Engineering 1, 57-94.

[Dum88] D. Dumitrescu, Hierarchical pattern classification, Fuzzy Sets and Systems 28 (1988), 145-162.

[Hat01] Hathaway R. J., Bezdek J. C., Fuzzy C-Means Clustering of Incomplete Data, IEEE Trans. Systems, Man, Cybernetics, Part B: Cybernetics, 31, 5 (2001), 1062-1071.

[She96] Shepperd M., Schofield C., Kitchenham B., Effort Estimation Using Analogy, Proceedings of the 18th International Conference on Software Engineering, Berlin, 170-178.

[Pop95] Pop H. F., Supervised fuzzy classifiers. Studia Universitatis Babes-Bolyai, Series Mathematica 40, 3 (1995), 89-100.

[Wat67] Watanabe S., Haven H., Karhunen-Loeve expansion and Factor analysis. Theoretical remarks and applications, in Transactions of the Fourth Prague Conference on Information Theory, Statistical Decision Functions, Random Processes, Prague 1967.

Department of Computer Science, Babeş-Bolyai University, 1 M. Kogălniceanu St., 400084 Cluj-Napoca, Romania
E-mail address: mfrentiu@cs.ubbcluj.ro

Department of Computer Science, Babeş-Bolyai University, 1 M. Kogălniceanu St., 400084 Cluj-Napoca, Romania
E-mail address: hfpop@cs.ubbcluj.ro