

Determining the Incremental Worth of Members of an Aggregate Set through Difference-Based Induction

Avelino J. Gonzalez
University of Central Florida
PO Box 162450
Orlando, FL 32816-2450
ajg@ece.engr.ucf.edu

Sylvia Daroszewski
Harris Corporation
Palm Bay, FL

Howard J. Hamilton
Dept. of Computer Science
University of Regina
Regina, Canada
hamilton@cs.uregina.ca

Abstract

Calculating the incremental worth or weight of the individual components of an aggregate set when only the total worth or weight of the whole set is known is a problem common to several domains. In this article we describe an algorithm capable of inducing such incremental worth from a database of similar (but not identical) aggregate sets. The algorithm focuses on finding aggregate sets in the database that exhibit minimal differences in their corresponding components (referred to here as *attributes* and their *values*). This procedure isolates the dissimilarities between nearly similar aggregate sets so that any difference in worth between the sets is attributed to these dissimilarities. In effect, this algorithm serves as a mapping function that maps the makeup and overall worth of an aggregate set into the incremental worth of its individual attributes. It could also be categorized as a way of calculating interpolation vectors for the attributes in the aggregate set.

Introduction

Knowing the relative contribution of individual components of an aggregate set in terms of their worth (or cost, weight, etc.) to the total worth (or cost, weight, etc.) of the whole set can be important in domains such as engineering analysis, scientific discovery, accounting, and real estate appraisal. A slightly different but very similar problem is that of obtaining the incremental worth of such individual components. This task can be a difficult one when only the total worth of the whole set is known.

The work described in this article focuses on an approach to calculate the *incremental* worth of components of aggregate sets. This is possible when a database contains many aggregate sets that are homogeneous (i.e., have the same attributes), but dissimilar (i.e., have different values). An algorithm is formulated which finds aggregate sets in the database that exhibit minimal differences in their attributes' values. This procedure strives to isolate the single differentiating attribute/value combination of nearly similar aggregate sets.

1. Copyright 1998 American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

Any difference in worth between the sets can be attributed to this singular difference. For this reason, the algorithm is called the *difference-based induction algorithm*, or DBI algorithm for short.

Definition of Terms

An *aggregate set* represents a collection, assembly or grouping of member components which have a common nature or purpose. An aggregate set is made up of these aforementioned components. The term *attribute* will be used to signify a component of an aggregate set, and each attribute is said to have a particular *value*. A group of such attribute/value combinations can singularly describe a specific aggregate set.

The components of an aggregate set act as integral members, which individually add worth to the aggregate set. However, their individual contributions to the total worth of the aggregate set may not be explicitly known. We will use the term *worth* here generically to mean a real number representing the worth, weight, cost, price or any other similarly quantifiable factor.

An *aggregate set*, AG , is therefore defined as a pair consisting of a set containing its components (called *attributes*, A), and the total worth of the aggregate set, (*worth*). Therefore, the i^{th} aggregate set AG_i (of n total aggregate sets) can be represented by a pair as

$$AG_i = \langle A_i \text{ worth}_i \rangle.$$

A_i is further defined as the group of attributes for the set AG_i , such that for m attributes in AG_i :

$$A_i = \{a_{i,1} \ a_{i,2} \ a_{i,3} \ a_{i,4} \ a_{i,5} \ \dots \ a_{i,m}\}$$

where $a_{i,j}$ is the *name* of the j^{th} attribute of the i^{th} aggregate set.

Furthermore, each attribute, $a_{i,j}$, represents another pair depicting the *value* of the attribute ($v_{i,j}$), and its individual *worth* ($w_{i,j}$). The individual worth of an attribute is the amount that this attribute contributes to the overall worth of the aggregate set (worth_i). In the problems described here, $w_{i,j}$ is typically not known.

Care should be taken to not confuse the terms "value" and "worth". *Value* is the value inherent with that attribute, much like attribute-value pairs in frames or the

- i) Using the criteria above, select the next test attribute from the **Test_Attribute_List**.
 - ii) Assign the test attribute to the current node of the classification tree.
 - iii) Create branches corresponding to possible values of the currently selected test attribute.
 - iv) Distribute each aggregate set into the next level by assigning them to the corresponding branches, according to their value for the attribute being tested at the current node.
 - v) Delete the test attribute from the **Test_Attribute_List**.
 - vi) Create a new unlabeled node level at the end of branches generated.
 - f) Assign **critical_attribute** to the last node level created.
 - g) Create branches corresponding to possible values of the currently selected test attribute.
 - h) Distribute each aggregate set into the leaf level by assigning them to the corresponding branches, according to their value for the attribute being tested at the current node.
 - i) Prune the aggregate sets irrelevant for analysis through the heuristics described in Section 3.3.4
 - j) Apply paired-leaf analysis to determine the partial worth estimate for the critical attribute.
 - k) Compute the final worth for the critical attribute.
- 6. End**

A more detailed description of the system in pseudocode can be found in [Doroszewski, 1995].

Implementation and Evaluation of the Difference-Based Induction Algorithm

A prototype was built which implements the DBI algorithm. The domain used for the prototype was in residential property appraisal. This domain lends itself very well to this approach because determining the incremental worth of the specific attributes of a house (the aggregate set) has a significant impact on its appraised value (its overall worth). Large databases of sold houses are typically available to be used to determine the incremental worth of several attributes.

We developed a DBI algorithm prototype and extensively tested it on a database of 84 single-family houses sold during 1994 in the Alafaya Woods subdivision, Oviedo, Florida. Sales data was obtained from the Multiple Listing Service (MLS) database

The prototype will accept the entry of a feature of a house whose incremental worth is to be computed from the MLS database for that section of the city at that time. It will return a single number indicating what the market considers to be the incremental worth of that property feature when compared to one of lesser worth

Results of Prototype Evaluation

To evaluate its effectiveness, its results were compared to those developed manually by an expert for the same neighborhood area during the same period of time. The attributes compared were the number of bedrooms, the number of bathrooms, the living area, existence of a pool, existence of a garage, existence of a fireplace and age of house. The DBI prototype results were in the form of a range of values, something typically done in the appraisal business. The percent difference between the maximum and minimum of said ranges when compared to the corresponding ranges computed by the expert were as follows (the comments in parentheses represent the expert's evaluation of the comparison): Living area: 0-2.4% ("acceptable"); Bedrooms: 49-72% ("marginally acceptable"); bathrooms: 154-186% ("unacceptable"); garage: 0.3-49% ("acceptable"); swimming pool: 2.5-19% ("acceptable"); fireplace: 2.5-109% ("low end acceptable, high end not acceptable"); age of house: 20-42% ("acceptable"). The complete data and a more detailed discussion of results can be found in (Doroszewski, 1995).

Conclusion

The results obtained indicate that the procedure worked well with a relatively small database. However, unacceptable results were obtained for the number of bedrooms, fireplace, and the number of bathrooms. These discrepancies could be attributed to the limited number of houses populating the leaves used in the paired-leaf analysis.

References

- Doroszewski, S.G. 1995. Mining Metric Knowledge from Databases Using Feature-Oriented Induction. Master's thesis, Department of Electrical and Computer Engineering, University of Central Florida, Orlando, FL.
- Quinlan, J. R. 1983. Learning Efficient Classification Procedures and their Application to Chess End Games. In Michalski, R. S., J. G. Carbonell, and T. M. Mitchell, eds *Machine Learning: An Artificial Intelligence Approach*. San Mateo, California: Morgan Kaufmann.
- Quinlan, J. R. 1987. Decision Trees as Probabilistic Classifiers. In Proceedings of the Fourth International Workshop on Machine Learning, 31-37. Irvine, California.
- Quinlan J. R. 1993. *C4.5: Programs for Machine Learning*, San Mateo, California: Morgan Kaufmann,.