ELSEVIER

Computer Methods and Programs in Biomedicine 48 (1995) 39–44

computer methods and programs in biomedicine

User-requirements driven learning

Vincent Karthaus*[a], Helge Thygesen[b], Michael Egmont-Petersen[a], Jan Talmon[a], Jytte Brender[b], Peter McNair[c]

[a]*Department of Medical Informatics, University of Limburg, Limburg, The Netherlands*
[b]*Medical Informatics Laboratory Aps, Lyngby, Denmark*
[c]*Department of Clinical Chemistry, Hvidovre Hospital, Hvidovre, Denmark*

Abstract

This paper describes an approach for deriving classification knowledge from databases, taking into account user preferences. These preferences especially concern the trade-off between different kinds of costs and performance indicators of the classification scheme to be developed. We analyze what knowledge, provided by the user, can be used at various stages of the machine learning process to influence the development of the classifier. We restrict ourselves in this paper mainly to the generation of classification trees.

Keywords: User requirements; Classification system; KAVAS Projects

1. Introduction

It has been recognized that programs which can derive sequential classification schemes, are useful tools for developing strategies for data acquisition. The need for such tools was one of the main reasons for initiating the KAVAS (A1021) and KAVAS-2[1] (A2019) AIM projects. Among other things KAVAS-2 is aiming at the development of an integrated knowledge acquisition and validation tool, named KAVIAR[2]. This tool will assist the medical expert in his task of formalizing his own domain knowledge as well as in building classifiers suited to the domain of investigation. It will provide options for knowledge modelling and for building classification models from databases, among these an induction algorithm that generates classification trees.

Earlier experiments in the KAVAS project have identified the need for user guidance in the generation of classification models [1] and have outlined a preliminary solution [2]. In the following, we will describe the kind of requirements the user may have regarding the properties of the classifier to be built. From this description we will derive in more detail the approach for guiding the classification tree-building process of the induction algorithm in KAVIAR. We will demonstrate that during different stages of the development of the classifier we can use the user requirements in order to guide this learning process.

* Corresponding author.

[1] Knowledge Acquisition, Visualization and Assessment System.

[2] Knowledge Acquisition, VIsualization And Refinement.

SSDI 0169-2607(95)01658-G

2. User requirements for classification procedures in medicine

Economic pressure on the health care system requires that effective patient management procedures are provided against reasonable costs. During the diagnostic process the clinician often has the possibility to order a number of tests that will shed light on the problem at hand. Certain tests may be more powerful than others with respect to the diagnostic performance but they may have higher costs, either economical or ethical. For example, one will give preference to a physical examination over a transcutaneous or more invasive test as the risks of complications are higher in the latter type of tests. Therefore, a clinician is inclined to use cheap tests (in an economical and ethical sense) in the early phases of the diagnostic process. In later phases, when certain possibilities have been excluded or when a possible diagnosis has reached a sufficiently large likelihood, more expensive tests may be required. This kind of posing a preferred order on the possible tests, thus, is largely determined by the costs of the tests and by ethical considerations [2]. Unfortunately, the tests that provide most information are often the more invasive and more expensive tests. To cope with this problem, the clinician must decide on a trade-off between the costs involved in carrying out a particular test and the diagnostic performance of such a test.

A second issue is the quality or performance of the diagnostic procedure. Taking into account that a 100% correct classification is often not obtainable, it is the context in which the diagnostic protocol will be applied that defines what errors are acceptable. In a screening situation, one surely does not want too many false positive alarms. The predictive value of a positive finding has to be high. On the other hand, in a specialized clinic, one surely does not want to overlook a serious condition. So the predictive value of a negative finding has to be high. The situation is usually more complicated because we frequently deal with multi-class problems. For instance, in ECG analysis, mixing lateral and anterior infarct patterns is of less harm than mixing, for example, cases with hypertrophies with cases with infarct patterns. This shows us that we should give the user — the clinician — the opportunity to specify preferences to some classes (outcomes) over other classes during the development of the classifier.

3. An approach for user-requirements driven learning

The knowledge produced by the induction algorithm is represented in the form of a classification tree, where the nodes represent subsets of examples of the training set and the outgoing branches from a single node constitute conditions which involve one attribute, see Fig. 1. The conditions are mutually exclusive and collectively exhaustive, thus they induce a partition on the set of examples at the node.

The basic operation of the induction algorithm is to recursively divide the set of examples into subsets such that the distributions of cases over the different classes within the various subsets differ as much as possible. In our algorithm, we utilize the reduction in entropy as the guiding measure [3]. It is only the discriminatory power of the attribute values that is taken into account by this measure.

When we pose no restrictions on the partitioning process, we will end with a classification tree that discriminates perfectly among all examples in the training set, excluding duplicated examples which belong to different classes. Nevertheless, the predictive power of that tree may be small as it largely fits the noise in the training set.

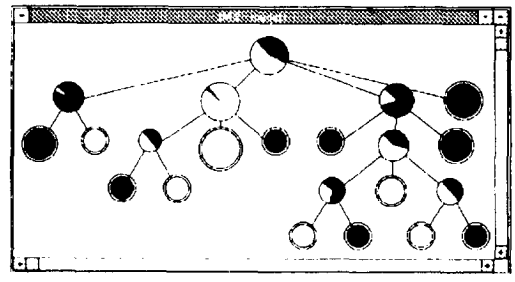

Fig. 1. Classification tree for discriminating between three classes.

To avoid overfitting, one can define requirements for the reliability of each of the partitionings. Earlier experiences with the algorithm revealed good classification performance of trees when we use the unreliability of the partitioning as a stopping criteria. However, as the decision to stop is made locally, there is no guarantee that an optimal decision is obtained. This situation could arise when no good splits were produced by node t but, due to additional information which became available through one or more 'bad' splits of t descendant nodes of node t contain good splits — we missed some opportunity to improve the tree. One can therefore adopt the strategy to build classification trees that overspecialize to the data in the training set, but from which the branches with low predictive values are pruned.

In our implementation of the induction algorithm, the user can set a variety of parameters for the classification tree as a whole as well as for each node in the tree. The user can, for example, determine that some attributes should not be considered in finding the optimal split or he could manually assign a class label to a leaf node. Varying these parameters as well as pruning the whole tree, leads to a set of closely coupled classification trees for the same domain. So in the end, the user may have a large series of classification trees. The final problem the user now faces, is how to select among these alternatives the best classification model suitable for his specific problem.

We propose that building classification trees is a three-stage process. Each phase in this process employs user requirements to direct the development of the final classifier. By this approach the classifier produced is more tailored to the user's expectations. The first stage deals with the generation of an initial set of classification trees. The second stage deals with pruning these trees, and finally, the third stage handles selecting among various alternatives.

3.1. The tree generation phase

In this phase of classifier development there are a few parameters, mainly addressing the internal behaviour of the induction algorithm, which can be adjusted by the user, for example, the choice between binary or n-ary splits or the method to handle missing values in the example cases. The primary issue here is the selection of the set of attributes which will be considered as possible tests. In our current version of the algorithm, the user has to make this selection manually. However, when the user has specified the costs of acquiring an attribute, one can rank the attributes with respect to those costs. (The model describing the cost structure could include attribute costs that depend on the situation, for example, higher costs for certain categories of patients, or which are dependent on whether other attributes have already been established). The basic problem now is to balance entropy reduction (the information content) against the costs of acquiring the attribute. This is not a trivial task. In the induction algorithim decisions on which test to use are made locally, while the total cost is based on the overall procedure. Furthermore, it will be difficult to specify the economical or ethical value of one bit of information. Instead, we propose to generate a family of reasonable trees, which will be evaluated and presented to the user for assessment.

What family of trees is produced will depend on the discriminatory power and the costs of the available attributes. At each step in the generation process, a node is selected for expansion and the attributes are ranked according to their entropy reduction. The tree is expanded using the best performing attribute. Again, as this decision is taken locally there is no guarantee that the resulting tree will be globally optimal. Therefore, we also create other expansions based on attributes that locally have a lower information content. So as to avoid expansions based on good performance but costly attributes the generated partial trees are sorted according to increasing costs. The cheapest tree is now a candidate for an additional expansion step. This way, each (partial) tree gives rise to a set of other (partial) trees. This process is repeated until the top N cheapest trees cannot be further expanded. The decision when to stop the search can either be based on the costs of the various trees or on their diagnostic performance after pruning (see below). These N trees will now be subjected to the next phase of the induction process.

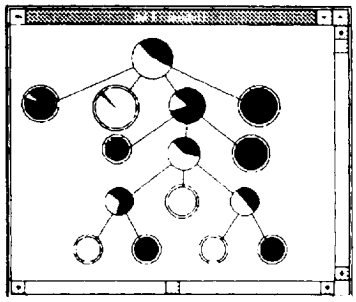

Fig. 2. Cost distribution for different misclassified and unclassified cases.

3.2. The pruning phase

Post-pruning of classification trees as described in the literature relies often on some statistical measure, the number of misclassifications and/or the complexity of the tree [4]. Programs that use post-pruning optimize this measure by cutting off branches until the overall performance cannot improve. Such approaches do not account for the fact that one type of misclassification may be worse than another or that a minimum performance for certain classes may be required.

We have developed a post-pruning method that is directed by user requirements. To make these explicit, the user has to specify minimum predictive values for each class as well as his preferences by assigning costs to the various misclassifications (see the cost matrix in Fig. 2). The relationships between these costs should reflect the user's assessment of the seriousness of each of the misclassifications. The specification of the cost matrix might in fact be a tedious task to perform. To relieve the user in specifying these costs, we make use of the property that the minimum predictive values are partially reflected by the cost matrix.

To each of the trees produced in the tree-generation phase we apply the following pruning scheme. Firstly, we assign a class label to all nodes in the tree such that the predictive values are not violated. In all other cases, we label the nodes as 'unclassified'. This labelling of nodes is based on the training set. Next, we apply to the tree a set of test cases. When the predictive values are not met for this test set we relabel nodes as unclassified. Next we simplify the tree by merging two sibling nodes if the costs of the resulting node do not exceed the sum of the costs of the original nodes or when the original nodes are equally classified (for binary trees this boils down to deleting a subtree). This procedure starts from the leaf nodes and is recursively applied until we reach the root of the tree.

After we have applied this procedure to each tree of the family created in the tree-generation phase, we get a new set of more simplified trees, which follows as close as possible the requirements of the user. (For an example see Fig. 3.)

3.3. The selection phase

After the tree-generation and pruning phases, the user has to choose among the available classification trees. This selection process can even be extended by including other classification models as well. The approach described here has initially been developed to select among a series of neural nets produced with the neural network generator in KAVIAR. The selection task consists of balancing some, possibly many, relevant criteria against each other in order to distinguish acceptable from unacceptable classification models. So the user is faced with a Multi Criterion Decision Making (MCDM) problem. This MCDM problem may be approached by several strategies. It is common to see the selection process as an interaction between a user (Decision Maker) and a computer (ANalyst) [5]. Such an interaction could

Cost matrix

Database

Model \ Database	A	B	C
A		6	17
B	7		4
C	20	8	
Unclass.	2	5	2

Fig. 3. Pruned classification tree using the cost matrix in Fig. 2.

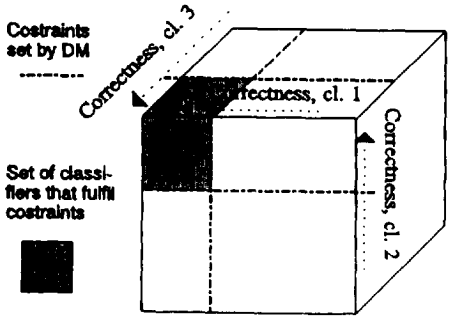

Fig. 4. Plane covering all classifiers.

either be directed by the Decision Maker (DM directed) or by the ANalyst (AN directed). In the first case the user is directly responsible for the actual choice whereas in AN-directed approaches, the user enters his preferences into the computer, which then selects the best alternative.

We decided to design and implement both a DM-directed and an AN-directed selection algorithm. In the DM-directed approach the user has to define a set of constraints, currently limited to class-conditional correctness and coverage measures [6]. All classifiers that do not fulfil these constraints are discarded. The DM can now browse through the remaining tables (shaded grey in Fig. 4). The user can then make the constraints more rigorous such that the set of satisfactory classifiers is reduced. However, if the DM finds the number of satisfactory classifiers acceptably low, it is now up to him to rank the classifiers by different performance criteria, and finally he is to select the best classifier. So the DM does not depend on the computer's assessment of the different criteria and the DM does not have to quantify the trade-offs but is himself responsible for the selection process.

Our AN-directed approach is directly based on utility theory [5]. The user should express his preferences by assigning costs to the various misclassifications (see the pruning phase of the tree building process). Then the AN (the computer) calculates a 'disutility' measure for each classifier based on its contingency table and the cost matrix (Fig. 2) using the following formula:

$$\mathrm{Du} = \sum_{i=1}^{r} \sum_{j=1}^{c} Table_{ij} \cdot Cost_{ij}$$

The classifiers are ranked according to increasing *Du*. The DM can investigate the sensitivity of the result by systematically varying the costs of misclassifications, [see for example, 7,8].

4. Summary

We have discussed how we could incorporate user requirements into the development process of classification models, in particular classification trees. These requirements concern the costs of attributes (ethic and/or economic) as well as the costs of various misclassifications. We have separated the development process into a tree generation, pruning and selection phase. In each phase user requirements are used to reduce or simplify the set of relevant classifiers. This way, the user has a stronger grip on the development of the classifiers. In addition, the final classifier(s) is best suited to the user's application.

Acknowledgements

This work was partly funded by the Commission of the European Communities under the AIM Exploratory Act (KAVAS (A1021) Project) and the AIM Telematics in Health Care Program (KAVAS-2 (A2019) Project). The development of the MCDM approach was done by Michael Egmont-Petersen when he was employed as an industrial PhD student at CRI Birkerød, Denmark, where he was supported by a bursary EF-348 from the Danish Academy of the Technical Sciences. We gratefully acknowledge Thomas Schiøler for valuable discussions about this work.

References

[1] J.L. Talmon, P. Braspenning, J. Brender and P. McNair, Machine learning in data rich domains: some experiences from the KAVAS project, Proceedings of the 3rd Conference on Artificial Intelligence in Medicine Europe, pp. 283–293 (Maastricht 1991).

[2] P. McNair, V. Karthaus, J. Talmon, H. Thygesen, T. Schiøler and J. Brender, KAVAS's conditioning of the induction algorithm, Proceedings of the 4th Conference on Artificial Intelligence in Medicine Europe, pp. 425–428 (Munich, 1993).

[3] J.L. Talmon, A multiclass nonparametric partitioning algorithm, Pattern Recognition Letters 4 (1986) 31–38.
[4] J. Mingers, An empirical comparison of pruning methods for decision tree induction, Machine Learning 4 (1989) 227–243.
[5] P. Bogetoft and P. Pruzan, Planning with multiple criteria: investigation, communication and choice (North-Holland, 1991).
[6] M. Egmont-Petersen, J.L. Talmon, J. Brender and P. McNair, On the quality of neural net classifiers, AIM 1994 (accepted for publication).
[7] A.M. Geoffrion, Solving bicriterion mathematical programs, Oper. Res. 15 (1967) 39–54.
[8] R.M. Soland, Multicriteria optimization: a general characterization of efficient solutions, Decis. Sci. 10 (1979) 26–38.