A Tool for Automatic Creation of Rule-Based Expert Systems with CFs

Ioannis Hatzilygeroudis and Konstantinos Kovas

Department of Computer Engineering & Informatics, University of Patras
GR-26500 Patras, Greece
{ihatz,kobas}@ceid.upatras.gr

Abstract. This paper introduces a tool, namely ACRES (Automatic CReator of Expert Systems), which can automatically produce rule-based expert systems as CLIPS scripts from a dataset containing knowledge about a problem domain in the form of a large number of cases. The rules are created via a simple systematic approach and make use of certainty factors (CFs). CFs of same conclusions can be combined either using the MYCIN method or a generalization of MYCIN's method. This latter method requires calculation of some weights, based on a training dataset, via the use of a genetic algorithm. Creation of an expert system is outlined. Small scale experimental results comparing the above methods with each other and a neural network are finally presented.

Keywords: expert systems, certainty factors, unsupervised learning, MYCIN, genetic algorithm, CLIPS.

1 Introduction

A lot of datasets are available nowadays, containing known instances from various domains (e.g. UCI Machine Learning Repository). Effort is required to retrieve information from these datasets and use them as a knowledge base for predicting new instances. A tool is therefore needed that can automatically turn these raw data to rules that can classify new data accordingly using a model of uncertainty. Existing efforts at providing such automated tools are mostly commercial and thus targeted to professionals or knowledge engineers (e.g. [1]), making the process difficult for intermediate users or domain experts that want to make use of existing knowledge in datasets. The uncertainty models implemented are also not efficient in many cases or lack the ability to adjust to the requirements of each individual problem. The main goal of this work is to introduce a tool (ACRES), which can make use of datasets and other available sources of knowledge, in order to easily create expert systems that use that knowledge to cope with new instances of a problem. A generalized version of the MYCIN certainty factors (CFs) is used as the way to represent uncertainty, allowing optimization of the expert system. In section 2 we present existing tools for the creation of expert systems. In section 3 we present certainty factors as a way to represent uncertainty in expert systems and present the two methods we implemented. The way the automatic generation of expert systems from datasets can be achieved is described in Section 4. In section 5, an example creation of an expert system is outlined. In

H. Papadopoulos, A.S. Andreou, and M. Bramer (Eds.): AIAI 2010, IFIP AICT 339, pp. 195–202, 2010.

section 6, some experimental results are presented. Finally, section 7 concludes and discusses future works.

2 Related Work

One of the most well-known tools for developing rule-based expert systems is CLIPS (C Language Integrated Production Systems) environment from NASA [4]. JESS (Java Expert System Shell), its Java counterpart, has also received great interest [7]. Expert System Creator [9] is a commercial software tool for the development of expert system based applications, representing domain knowledge using rule sets, decision tables or classification trees. It is mainly targeted to professionals, domain experts or knowledge engineers and less for simple users. EXSYS [1], an also commercial Expert System Development Software, is easier to use and provides many different uncertainty models to choose from. Our proposed tool focuses on the automatic generation of expert system knowledge bases from existing datasets with the minimum required effort from the user. At the same time it offers the ability to optimize the output expert system taking into account specific requirements or priorities set by the developers-users.

3 Expert Systems and Certainty Factors

3.1 Combining Conclusions with Certainty Factors

MYCIN [2] was a medical expert system developed in Stanford University in the early 1970s. It was the first one that introduced Certainty Factors (CFs) as a way to represent uncertainty when a conclusion is made by a rule. Although CFs have some problems [8], they still remain a simple and useful way of dealing with uncertainty.

CFs are associated with a simple computational model that permits to estimate the confidence in conclusions being drawn. A CF is a number between –1 (definitely false) and +1 (definitely true), which measures the expert's belief (positive number) or disbelief (negative number) to a conclusion. However, usually CFs that are positive numbers between 0 (definitely false) and 1 (definitely true) are resulted, due to the nature of most problems. In these cases, any CF less than 0.5 indicates disbelief, whereas any CF equal to or greater than 0.5 indicates belief in a conclusion. Given that CFs are positive, when we have the following rules with the same conclusion and CF1, CF2 respectively and they are fired, the combined certainty CF for conclusion, according to MYCIN theory, is given by the formula:

$$CF = CF_1 + CF_2 (1\text{-}CF_1) = CF_1 + CF_2 - CF_1 * CF_2 \quad (1)$$

3.2 Generalizing Certainty Factor Computation

The above formula didn't give satisfactory results in many cases. So in the expert system PASS [3], a generalized version of (1) was introduced:

$$CF = w_1 * CF_1 + w_2 * CF_2 + w * CF_1 CF_2 \quad (2)$$

where w1, w2 and w are numeric weights that should satisfy the following equation:

$$w_1 + w_2 + w = 1 \tag{3}$$

to assure that $0 \bullet CF \bullet 1$.

To use formula (2), however, the weights w_1, w_2, w should be first determined. In PASS, statistical data about the problem was used, as a training data set to determine the weights by hand. The tool presented in this paper computes the above weights automatically, utilizing a genetic algorithm.

4 Automatic CReation of Expert Systems (ACRES)

4.1 Rule and CF Generation

One of the main functions of the tool is to convert raw data in a dataset to a set of rules. We consider that each instance of the data set contains discrete values corresponding to attributes related to a problem domain. One of the attributes represents the class (or output) attribute. We consider binary class attributes, i.e. datasets referring to two classes (A and B). So, class attribute takes one of two values, e.g. yes-no or true-false. We distinguish between positive instances (those belonging to the class 'yes' or 'true') and negative instances (the rest ones). Rule and CF generation is then achieved by the following process:

1. Cluster instances in groups, so that each group contains instances that have identical values for all non-class attributes.
2. From each such group produce one rule with as conditions the attribute-value pairs of the instances and as conclusion the class attribute-value pair (where the value is the one of the positive instances).
3. Associate with each rule a CF defined as

   $$CF = n_p/N$$

   where n_p is the number of the positive instances in the group and N the number of all instances. That is, a CF is defined as the frequency of the positive instances in the group.

4.2 Expert System Creation

ACRES can produce one CLIPS-based expert system (ES) from a given data set. The process is as follows:

1. Specify the attributes and the class attribute of the data set.
2. Divide attributes in two possibly (typically) overlapping groups.
3. Extract from the dataset two subsets, corresponding to the two attribute groups. Each subset includes instances having values for the attributes of the corresponding group and the class attribute.
4. Produce rules with CFs separately from the two subsets using the process in Section 4.1.
5. Produce the rule implementing computation of the combined CF, based on formula (2) above.
6. Produce the basic expert system
7. Determine the weights w_1, w_2 and w of formula (2).

The rationale behind the division of the attributes and the dataset in two groups/subsets is the need to have two stages of reasoning or two alternate ways of reasoning for the same conclusion, from which the need for CFs combination comes.

In step 7, if we want to produce a MYCIN-like expert system, the system sets $w_1 = w_2 = 1$ and $w = -1$. If not, the genetic algorithm is used.

4.3 Genetic Algorithm

We use a simple genetic algorithm to determine the best values for weights w_1, w_2 and w, since it is an optimization problem. An initial population of possible solutions, called *atoms* (or chromosomes or genomes), is created. Then the atoms of the population are evaluated via a fitness function and the best of them are selected to pass to the next generation after having been processed through two processes, i.e. crossover and mutation, which resemble corresponding genetic processes. The algorithm stops when any of the set termination criteria is met.

In such an algorithm, there are two crucial design decisions: the representation scheme of an atom and the fitness function. In our case, an atom is a representation of the values of weights w_1 and w_2, since w can be calculated as $w = 1 - (w_1 + w_2)$. The fitness function should evaluate how well the represented weight values in an atom affect the effectiveness of the expert system under creation. Usually, three metrics are used for evaluating classification-oriented expert systems: *accuracy* (abbr. *acc*), *sensitivity* (abbr. *sen*) and *specificity* (abbr. *spec*) [6, 7]. So, the fitness function should be based on them. Although accuracy is the most important, sensitivity and specificity should be taken into account too. Also, sometimes a system is required to give more emphasis to one of them. In our GA implementation, the fitness function is defined as

$$fitness = w_{ACC} * acc + w_{SEN} * sen + w_{SPE} * spec + w_{BAL} * bal \quad (3)$$

where w_{ACC}, w_{SEN}, w_{SPE} and w_{BAL} are user-defined factors and '*bal*', which is an abbreviation for '*balance*', is defined by the following expression:

$$bal = 1 - |sen - spec| \quad (4)$$

which represents how balanced sensitivity and specificity are.

With the above definition we can define a variety of fitness functions, depending on which metric and in what degree we consider as more significant for the under creation expert system.

To compute fitness for each atom of the population, we need to evaluate the expert system under construction. So, the CLIPS engine is embedded in our system and is called to run the constructed expert system for all instances of the dataset. Based on the results we compute the values of the three metrics and then the fitness for each instance.

4.4 Systems Comparison

As is stated above, ACRES can produce two types of expert systems with CFs, one of MYCIN type and one of WEIGHTED type. The second type uses the generalized way of CF combination. This is related to the *expert system creation mode* of the tool.

However, apart from that, ACRES give the possibility to compare the two types of systems. So, in its *method comparison mode*, the system creates two similar systems

from the same dataset, one of MYCIN and one of WEIGTHED type, and compares them. Comparison process is as follows:

1. Split the dataset in two sets, a training and a test set.
2. Produce the basic expert system using the training set as the data set.
3. Produce the two expert systems (the MYCIN-type and the WEIGHTED-type).
4. Evaluate the two systems (i.e. compute *acc*, *sen* and *spec*) using the test set.
5. Repeat steps 1-4 k times (k is defined by the user).
   Repetition of steps 1-4 actually implements a k-fold validation process.

4.5 Implementation Issues

Both modes were implemented as console applications with C++, integrating the CLIPS library to load the expert systems created and the GALIB library for the genetic algorithm. The system uses a population size = 15, p_c = 0.7, p_m = 0.05 and terminates after 50 generations. This last parameter was chosen after having made a number of experiments. The graphical user interface was developed with Visual C++.

5 Example Use of the Tool

This section presents an example use of the presented tool, using a dataset from the UCI repository [8]. The breast cancer dataset was chosen, containing 286 instances of patient instances.

There are 10 variables in the dataset. The user should prepare two files. The one is the dataset, as provided in the UCI repository. In our case, it is the "breast.data" dataset file. The second file is the variables file ("VariablesNames.txt"), which contains the names of the attributes of the problem related to the dataset. The first variable is the class variable. In our case, it has two classes: "no-recurrence-events" and "recurrence-events".

In Fig. 1 the graphical user interface is shown, in the expert system creation mode. The user can browse for the above files. After loading them, he/she can choose the output/class variable. Next the user specifies two variable groups for which rules will be created. The way this is done is something that requires some knowledge about the domain (an expert's advice would help here). The groups can have common variables. Then, he/she can choose to create a MYCIN type system and/or a WEIGHTED type one. If the latter type is chosen, the GA parameters for the fitness function should be specified.

In the method comparison mode we can evaluate the two methods for combining conclusions. The user can specify the train/test datasets ratio and the cross validation folds. We've used 1/3 for the test/train ratio and 3-fold cross validation. After pressing the "Compare Methods" button the results of the evaluations of the two systems are depicted. Based on the results one can decide which of the methods is more appropriate for the problem and switch to the "expert system creation" mode. In our example, the WEIGHTED version of the expert systems does better.

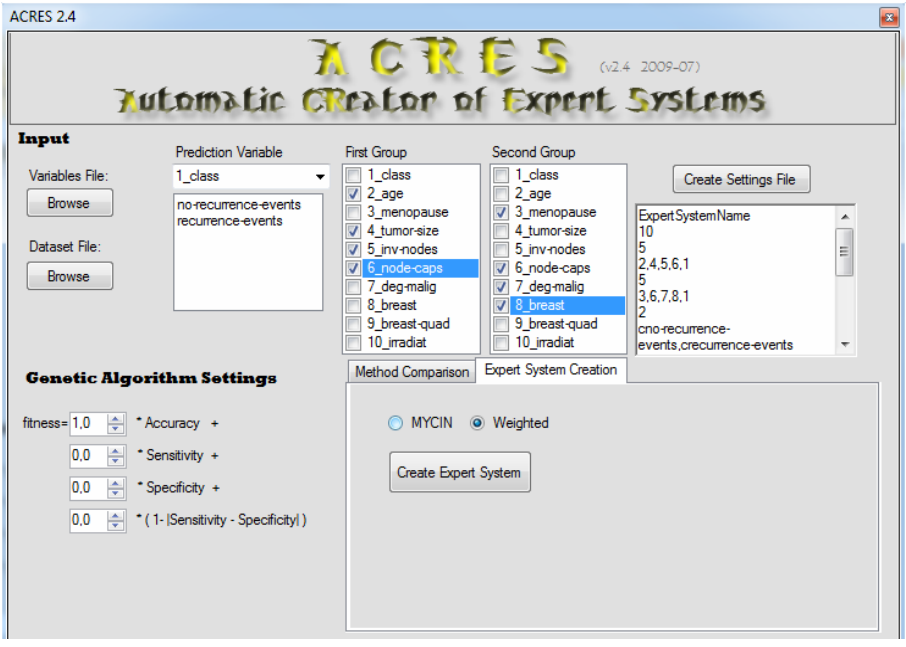

Fig. 1. ACRES Graphical User Interface

The expert system is created as a CLIPS program file. The program consists of four template definitions. The first template, named 'data', has as slots the variables of the dataset. The next two templates, named 'result1' and 'result2', respectively keep the results (classes and corresponding CFs) of the two reasoning stages for the two classes. The fourth template, named 'final' includes the final results from the combination of two different stage rules. The program also includes two groups of rules, one for each group/stage of variables/reasoning. Finally, it contains a rule that combines the results of two different stage rules. In Table 1, two rules, one from the first group and one from the second group of rules are presented, whereas Table 2 shows the final combination rule for the WEIGHTED version of the expert system (where w_1= 2.72175, w_2= 0.666972, w= -2.38872).

Table 1. Representative rules from the two rule groups

```
(defrule group1_1
 (data (age 40-49)
       (menopause premeno)
       (tumor-size 20-24))
  =>
 (assert (result1
           (no-recurrence-events
                        0.823529)
           (recurrence-events
                0.176471))))
```

```
(defrule group2_10
 (data (inv-nodes 3-5)
       (node-caps yes)
       (deg-malig 2))
  =>
 (assert (result2
           (no-recurrence-events
                        0.583333)
           (recurrence-events
                0.416667))))
```

Table 2. Rule for combination of results

```
(defrule ResultCombination
   (result1   (no-recurrence-events ?r1_0)
                    (recurrence-events ?r1_1))
   (result2   (no-recurrence-events ?r2_0)
                    (recurrence-events ?r2_1))
   ?y1<-(result1 (no-recurrence-events ?r1_0)
                    (recurrence-events ?r1_1))
   ?y2<-(result2 (no-recurrence-events ?r2_0)
                    (recurrence-events ?r2_1))
=>(bind ?r_0 (+   (* ?r1_0 2.72175) (* ?r2_0 0.666972)
                    (* ?r1_0 ?r2_0 -2.38872) ) )
  (bind ?r_1 (+   (* ?r1_1 2.72175) (* ?r2_1 0.666972)
                    (* ?r1_1 ?r2_1 -2.38872) ) )
  (assert (final (no-recurrence-events ?r_0)
                    (recurrence-events ?r_1)) )
  retract ?y1
  retract ?y2)
```

The CLIPS file can then be loaded in the CLIPS Expert System shell and make predictions about new instances of the problem.

6 Experimental Results

We used ACRES to produce two expert systems, one of MYCIN-type and the other of WEIGHTED-type) for predicting the success (or failure) of a technical high school student to the National exams in Greece. We used a real dataset consisting of 373 instances (student records). There were five attributes plus the class attribute. We also used WEKA [6] to produce a neural network for the same target. We used the same training and test sets for all systems and a 5-fold cross validation.

From Table 3, where results are presented, it is concluded that our ACRES weighted method of creating expert systems is comparable and in cases better than neural networks method. Additionally, construction of such systems in ACRES is much easier and controllable.

Table 3. Comparison of methods

METRIC	MYCIN	WEIGHTED	NN
acc	0.7334	0.8152	0.803
sen	0.7446	0.5639	0.630
spe	0.7291	0.9095	0.869

7 Conclusions and Future Work

We present a tool that can be used to easily create an expert system from existing knowledge stored as instances in a dataset. The tool introduces a method for creating rule-based expert systems with CFs in a CLIPS-based format. The method results in a two stages rule-based reasoning with CFs whose results are combined using a generalization of MYCIN's policy. The tool also allows for comparison of the two types of

systems, the MYCIN-based one and the WEIGHTED one, which is based on the generalized formula for CF propagation. Experimental results are promising.

There are however a number of possible improvements that are opportunities for further work. For example, the way the variable groups are selected is currently something completely assigned to the user. A way to provide him with a tool that can detect dependency formations between the dataset variables would be helpful. Also, regarding the input dataset, the tool requires that its variables have discrete values. If they don't, the user must take care of discretizing them. So, a facility helping towards this target is desirable. A more advanced solution to this would be the use of fuzzy logic combined with CFs (in a FuzzyCLIPS style) in the produced rules.

References

1. Awad, E.: Building Knowledge Automation Expert Systems with Exsys Corvid. Exsys Inc. (2003)
2. Buchanan, B.G., Shortliffe, E.H.: Rule-Based Expert Systems. In: The MYCIN Experiments of the Stanford Heuristic Programming Project. Addison-Wesley, Reading (1984)
3. Hatzilygeroudis, I., Karatrantou, A., Pierrakeas, C.: PASS: an Expert System with Certainty Factors for Predicting Student Success. In: Negoita, M.G., Howlett, R.J., Jain, L.C. (eds.) KES 2004. LNCS (LNAI), vol. 3213, pp. 292–298. Springer, Heidelberg (2004)
4. Culbert, C., Riley, G., Donnell, B.: CLIPS Reference Manual, vols. 1-3. Johnson Space Center, NASA (1993)
5. Frank, A., Asuncion, A.: UCI Machine Learning Repository. University of California, School of Information and Computer Science, Irvine, CA (2010), http://archive.ics.uci.edu/ml
6. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA Data Mining Software: An Update. SIGKDD Explorations Newsletter 11(1), 10–18 (2009)
7. Hill, F.: Jess in action: rule-based systems in Java. Manning Publishing (2003)
8. Hackerman, D., Shortliffe, E.: From certainty factors to belief networks. AI in Medicine 4, 35–52 (1992)
9. Pop, D., Negru, V.: Knowledge Management in Expert System Creator. In: Scott, D. (ed.) AIMSA 2002. LNCS (LNAI), vol. 2443, pp. 233–242. Springer, Heidelberg (2002)