Generalisation of Uncertain Concepts

Jim F. Baldwin, Carla Hill, Trevor P. Martin
Department of Engineering Mathematics, University of Bristol
Queens Building, University Walk, Bristol BS8 1TH, UK
Phone: +44-117-9287754, Fax: +44-117-9251154
email:{Jim.Baldwin,Carla.Hill,Trevor.Martin}@bris.ac.uk

ABSTRACT: We propose an approach to extend inductive logic programming (ILP) to cater for uncertainties in the form of probabilities and fuzzy sets. A corresponding algorithm for induction of Fril (a support logic programming language) rules which involve probabilistic uncertainty is also described. Given probabilistic examples for a target relation and the background concepts this algorithm generates a decision tree consisting of one branch which is directly translated into a Fril rule. This rule is a generalisation of the examples in the sense that it can, together with the background knowledge, explain the problem given by the examples. The approach is illustrated on a simple relational concept, friends(X,Y). Finally on-going current work is described which deals with enhancing the algorithm to handle fuzzy concepts for the background knowledge as well as for the target predicate.

KEYWORDS: Inductive Logic Programming, Support Logic, Generalisation of Clauses, Decision Trees, Fuzzy Sets, Mass Assignment Theory

INTRODUCTION

In inductive logic programming (ILP) logic programs are induced from positive and negative examples and some background knowledge. These programs or hypotheses together with the background knowledge can explain the examples, i.e. they are more general than the given examples, Muggleton (1994). In order to model uncertain and vague concepts we are working on extending ILP, which is currently limited to the classical bivalent logic, with methods that deal with uncertainties in the form of probabilities and fuzzy sets. In this paper we address the problem of inducing Fril rules for databases which consist, in contrast to classic ILP, of probabilistic examples, i.e. the examples for the target concept as well as for the background concepts are given with certain probabilities.

There exist two forms of ILP algorithms, one searches from general to specific, e.g. FOIL, Quinlan (1990, 1995), and the other one from specific to general, e.g. CIGOL, Muggleton (1992), for possible hypotheses. We chose a general-to-specific approach similar to FOIL where we consider all given examples at one time. Like FOIL we start with the most general rule, i.e. ((target relation)):((0 1)(0 1)). This means that the target relation is true with a probability between zero and one, which is always true. As in FOIL a decision tree is built which consists of one branch by adding concepts from the background to the body of the rule, i.e. to the conditions. These concepts are selected according to a decreasing weight, which expresses the importance of the concepts in order to explain the target concept. But the probabilistic database confronts us with some additional difficulties. First of all we need rules which can explain probabilistic examples. Therefore we use basic Fril rules where the rules are enhanced with two support pairs which represent the probability of the head, i.e. the conclusion, given the body and the probability of the head given not the body. In our approach we minimise the sum of the squared errors between the given and the calculated probabilities for the target relation in order to determine the support pairs. The used predictor function is simply the middle of the calculated interval, which we find with Jeffrey's rule, Baldwin (1995). Hence with the resulting support pairs for the induced rule the predicted probability lies as close to the original given probability as possible.

Another difficulty when working with probabilities is that the algorithm cannot be driven by the distinction between positive and negative examples. In our case all examples are treated as the same. Unlike FOIL we cannot stop adding literals to the body, i.e. conditions, when the rule stops matching any negative examples. Using probabilities the matching between the rule and the examples always provides us with a result, i.e. a probability, and therefore we have to find another stopping criterion. We have invented an error measure for a rule which is calculated as the sum of the absolute differences between the given and the calculated probability of the target relation for all examples. After adding a new literal we compare this measure with the one of the previous rule. If it increases we take the previous rule as the

resulting rule which is the best possible generalisation of the given examples. Otherwise we go on with the algorithm, i.e. add concepts to the body of the new rule.
In our current work we enhance this algorithm in order to handle fuzzy concepts for the target relation as well as for the background predicates. Using the mass assignment theory, Baldwin (1995), it is easy to convert membership values in probabilities, which allows us to use the same method as before. Additional problems occur if we consider more than one fuzzy set on each universe, for examples for a background concept. We then have to build decision trees with more than one branch, which makes the algorithm much more complicated.

BACKGROUND

Inductive logic programming is the intersection of inductive (machine learning) and logic programming, Muggleton (1994). Given some positive and negative examples of a target relation or predicate and some background knowledge that, on its own, is not able to explain the positive examples, a hypothesis is induced such that all positive and no negative examples can be logically derived from the hypothesis and the background knowledge. ILP algorithms provide several advantages over classical machine learning algorithms such as neural networks including: Their outputs are rules that are easily understood by people. They are relational-based and can therefore describe concepts, which cannot be easily described in an attribute-value language. Also background knowledge can be accommodated.
Fril is an Artificial Intelligence logic programming language that allows both probabilistic uncertainties and fuzzy sets to be included, Baldwin (1995). Fril rules are quantified by support pairs representing conditional probabilities and predicate arguments can be instantiated to fuzzy sets, Zadeh (1975/76). We can derive inference rules for Fril using mass assignment theory combined with Jeffrey's rule, Jeffrey (1965). Fril provides a natural environment in which ILP can be extended with uncertainties: In the proposed approach the induced models are expressed in term of probabilistic if-then rules where the variables can be fuzzy sets.
A fact then is represented in the form:

$$((\text{fact})) : \Pr(\text{fact}).$$

Rules consist of a head, i.e. the conclusion, and a body, i.e. the conditions. Each rule is also provided with two support pairs where the first pair represents the probability of the head given the body and the second pair represents the probability of the head given not the body. Hence a rule is of the form:

$$((h)(b)) : ((x_1\ x_2)(y_1\ y_2))$$

where $\Pr(h \mid b) \in [x_1,x_2]$ and $\Pr(h \mid \neg\, b) \in [y_1,y_2]$.
When we use a rule of this kind to calculate the probability of the head for a particular object with known probability for the body, Jeffrey's rule provides us with an interval:

$$\Pr_{calc}(h_o) \in [x_1 \Pr(b_o) + y_1 (1-\Pr(b_o)) , x_2 \Pr(b_o) + y_2 (1-\Pr(b_o))].$$

Here we use the middle of the interval as the predictor, i.e. the calculated probability is given by:

$$\Pr_{calc}(h_o) = 0.5 * (x_1 \Pr(b_o) + y_1 (1-\Pr(b_o)) + x_2 \Pr(b_o) + y_2 (1-\Pr(b_o))). \quad (1)$$

INDUCTION OF FRIL RULES WITH SUPPORT PAIRS

The given database consists of examples for the target concept and the background concepts where probabilities for the concepts are given for all examples. We can represent such a database as a table:

examples	Pr(Target concept t)	Pr(background concept c^1)	Pr(background concept c^2)
1	0.2	0.3	0.9
2	0.7	0.8	0.2

Table I: Example table for a given database

We now want to find a rule for the target concept that explains these examples.
As a first step we calculate weights for all background concepts which express the average importance of these concepts in order to explain the target concept:

$$w(c^k) = 1/N \sum_{\text{examples } i} Pr(t_i) * Pr(c^k_i),$$

where N is the number of given examples. Because a high weight corresponds to a high information content, we add the background concepts in order of decreasing weights to the body of the rule. Using this selection criterion we want to achieve short rules.

CALCULATION OF SUPPORT PAIRS

Our goal is to find support pairs such that the induced rule builds the best possible generalisation of the given examples, i.e. the predicted probabilities lie as close to the given probabilities as possible. Once we added a literal or concept to the body of the rule we receive a new rule with a new body and unknown support pairs:

$$((h)(b)) : ((x_1\ x_2)(y_1\ y_2)).$$

In order to determine the four variables we minimise the squared error between the given and the calculated probabilities for the target concept using the new rule. As mentioned above we use the middle of the interval as the predictor, giving the following formula for the squared error:

$$E^2 = \sum_{\text{examples}} [Pr_{\text{given}}(h_o) - Pr_{\text{calc}}(h_o)]^2$$

$$= \sum_{\text{examples}} [Pr_{\text{given}}(h_o) - 0.5\ (x_1\ Pr(b_o) + y_1\ (1\text{-}Pr(b_o)) + x_2\ Pr(b_o) + y_2\ (1\text{-}Pr(b_o)))]^2. \qquad (2)$$

The probability of the body for a particular object, $Pr(b_o)$, is the product of the probabilities of the single concepts in the body:

$$Pr(b_o) = Pr(b1_o) * Pr(b2_o) * Pr(b3_o) * \ldots$$

Evaluating the derivatives and setting these equal to zero gives us the conditions for the minimal squared error:

$$0 = -\sum_{\text{examples}} Pr_{\text{given}}(h_o)\ Pr(b_o) + 0.5\ x_1 \sum_{\text{examples}} Pr^2(b_o) + 0.5\ y_1 \sum_{\text{examples}} (1\text{-}Pr(b_o))\ Pr(b_o)$$
$$+ 0.5\ x_2 \sum_{\text{examples}} Pr^2(b_o) + 0.5\ y_2 \sum_{\text{examples}} (1\text{-}Pr(b_o))\ Pr(b_o) \qquad (3)$$

$$0 = -\sum_{\text{examples}} Pr_{\text{given}}(h_o)\ (1\text{-}Pr(b_o)) + 0.5\ x_1 \sum_{\text{examples}} Pr(b_o)\ (1\text{-}Pr(b_o))$$
$$+ 0.5\ y_1 \sum_{\text{examples}} (1\text{-}Pr(b_o))^2 + 0.5\ x_2 \sum_{\text{examples}} Pr(b_o)\ (1\text{-}Pr(b_o))$$
$$+ 0.5\ y_2 \sum_{\text{examples}} (1\text{-}Pr(b_o))^2 \qquad (4)$$

Notice that the derivatives for the two variables x_1 and x_2 yield the same equation (3), and the derivatives for the two variables y_1 and y_2 yield the same equation (4). Hence, all together, we just have two independent equations for four variables. Therefore we need two more equations which we obtain using an additional constraint. We consider the upper and the lower bounds of the given probabilities for the target concept and the body, respectively, over all given examples:

$$Pr_{\text{given}}(h_o) \in [h_l, h_u] \qquad \text{and} \qquad Pr(b_o) \in [b_l, b_u].$$

The additional constraint now requires that using the interval for the body as an input for the new rule, the output should be the interval for the head. We want to induce rules with high support pairs for the head given the body and low support pairs for the head given not the body. If this is not possible we choose another concept to add to the body.
Using this we get the additional equations:

$$h_l = x_1\ b_l + y_1\ (1 - b_l), \qquad (5)$$
$$h_u = x_2\ b_u + y_2\ (1 - b_u), \qquad (6)$$

where $x_1 \le x_2$, $y_1 \le y_2$, $x_1 \ge y_1$, $x_2 \ge y_2$, and $x_1, x_2, y_1, y_2 \in [0,1]$.
With all four equations we are now able to calculate the required support pairs.
If there exist no turning point of the squared error function for $x_1, x_2, y_1, y_2 \in [0,1]$, we check the extrema of the allowed values in order to find the minimum.

STOPPING CRITERION FOR ADDING MORE LITERALS

After the construction of a new rule, i.e. adding a new concept to the body and calculating new support pairs for this rule, we have to decide if the new longer rule is a better explanation of the examples than the previous one.
For each rule we calculate an error measure as the sum of the absolute difference between the given and the calculated probability of the target relation over all examples:

$$E = \sum_{examples} | Pr_{given}(h_o) - Pr_{calc}(h_o) | , \quad (7)$$

where the Pr_{calc} is determined by (1).
If this error increases after adding a new literal to the body, the previous rule provides a better description of the given examples. In this case we take the previous rule as the result because we can not find any better generalisation. All remaining background concepts have a lower weight, i.e. a lower information content, and will therefore lead to worse results. In the other case we choose the new rule and proceed with the algorithm.
If there are no literals left that could be added to the body, then the algorithm terminates.

ALGORITHM

1. Start with the most general rule R_0 = ((h)) : ((0 1)(0 1)) and calculate the error measure E_0
2. Calculate the weights for all body literals
3. Write the body literals according to their decreasing weights in an ordered list L_0
4. Take the first body literal from the list L_i and add it to the body of the rule R_i
5. Calculate the support pairs and the error measure E_{i+1} for the new rule R_{i+1}
6. If $E_{i+1} > E_i$, stop and result = R_i
   Else delete the first literal from the list L_i to get L_{i+1} and go to 4.
7. Stop when no literals left.

EXAMPLE

We chose a simple example to demonstrate how the method works. The small database shown in table II contains probabilistic examples for the target relation friends(X,Y) and for the background concepts likes(X,Y), likes(Y,X), similar_age(X,Y). We created the examples by choosing the probabilities for the background concepts arbitrarily, and afterwards calculating probability intervals for the target relation with the Fril rule:

((friends(X,Y))(likes(X,Y))(likes(Y,X)) : ((0.9 1)(0 0.1)).

We then picked a point probability from each interval by random.

	Pr(friends(X,Y))	Pr(likes(X,Y))	Pr(likes(Y,X))	Pr(similar_age(X,Y))
(Jill, May)	0.51	0.7	0.8	0.2
(Jill, Roger)	0.8	0.9	0.9	0.8
(May, Roger)	0.225	0.5	0.5	0.5
(Paul, Roger)	0.6	0.7	0.9	0.6
(Paul, May)	0.64	1	0.6	0.3
(Paul, Jill)	0.1	0	0.4	0.1
(Paul, Joe)	0.24	0.2	0.8	0.4
(Joe, Jill)	0.05	0.3	0	0.8
(Joe, May)	0.7	0.9	0.8	0.4
(Roger, Joe)	0.4	1	0.4	0.7
May, Alex)	0.49	0.8	0.6	0.3
(Alex, Joe)	1	1	1	1

Table II: Database for friends relation

Ordering the possible body literals according to their decreasing weights we get the list (likes(X,Y), likes(Y,X), similar_age(X,Y)). The algorithm stops after adding likes(Y,X) to the body, because the error measure increases when adding the next literal, similar_age(X,Y). Figure 1 shows the decision tree for this problem.

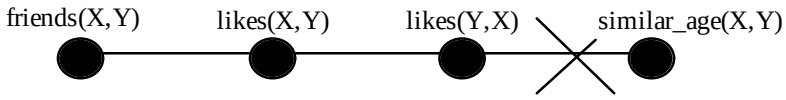

Figure 1: One-branch decision tree for friends-relation

Using the method explained above to calculate the support pairs, the resulting rule is:

((friends(X,Y))(likes(X,Y))(likes(Y,X)) : ((0.93 1)(0.05 0.06)).

We can read this rule like "X and Y are friends with a probability between 0.93 and 1 if they like each other. If they do not like each other the probability that X and Y are friends lies between 0.05 and 0.06.".
Table III shows the results we received using the constructed rule.

	Pr(friends(X,Y))	Pr_{cal}	$\lvert Pr_{given}-Pr_{calc}\rvert$
(Jill, May)	0.51	0.56	0.05
(Jill, Roger)	0.8	0.79	0.01
(May, Roger)	0.225	0.28	0.055
(Paul, Roger)	0.6	0.63	0.03
(Paul, May)	0.64	0.6	0.04
(Paul, Jill)	0.1	0.055	0.045
(Paul, Joe)	0.24	0.2	0.04
(Joe, Jill)	0.05	0.055	0.005
(Joe, May)	0.7	0.71	0.01
(Roger, Joe)	0.4	0.42	0.02
(May, Alex)	0.49	0.49	0
(Alex, Joe)	1	0.96	0.04

Table III: Results for friends relation

CURRENT WORK

At the moment we are developing an enhanced algorithm that can handle fuzzy concepts for the target relation as well as for the background concepts. The literals in the constructed rules can contain fuzzy sets in both the head and the body.
There exist different reasons why we like to use words with fuzzy semantics instead of precise definitions. We want to model humans who use imprecise concepts. Also we expect smoother solutions for problems with continuous outputs, because fuzzy methods automatically provide interpolation. In addition we hope to achieve greater compression of the knowledge bases resulting from the interaction of neighbouring words corresponding to fuzzy sets, which reduces the need for higher resolution. We are working on an inductive learning algorithm, which works in two steps:
First we partition the attribute universes into fuzzy sets representing the meaning of a set of words. For every element of a universe there exists a fuzzy set in the partition in which the element has membership one. The algorithm generates uniform linguistic partitions of trapezoidal fuzzy sets with given degree of overlap using the same methods like MA-ID3, Baldwin (1998).

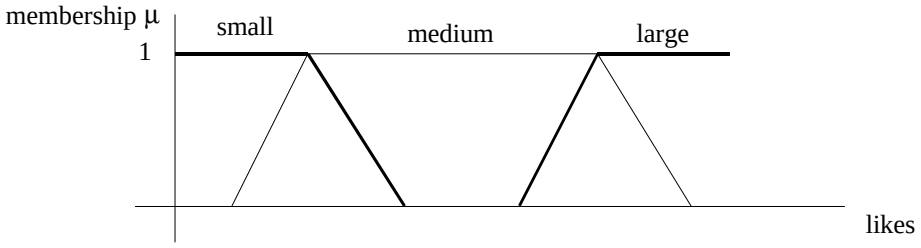

Figure 2: Fuzzy partition on likes

The database is transformed into a least prejudiced probability representation, i.e. each value is expressed as a fuzzy set on the words on the corresponding universe and this fuzzy set is then transformed into the least prejudiced distribution using the mass assignment theory, Baldwin (1995, 1996). Hence a database containing values for different universes as shown in table IV is converted into a probabilistic database like in table V.

examples	shoesize	height
1	39	1.70m
2	43	1.80m

Table IV: Numerical database

examples	shoesize			height		
	Pr(small)	Pr(medium)	Pr(large)	Pr(short)	Pr(medium)	Pr(tall)
1	0.5	0.5	0	0.25	0.75	0
2	0	0.5	0.5	0	0.75	0.25

Table V: Converted probabilistic database

The second step in the algorithm repeats generating decision trees for each single word on the target relation. We consider each label or word on the target relation as a separate target concept, e.g. small shoesize, and repeat building branches that explain a part of the subdatabase for this target concept. The subdatabase consists of all examples which have a given probability bigger than zero for the considered target concept. Every word on all background relations is treated as a background concept, which can be added to a branch if it leads to a support pair such that the support of the head given the body is bigger than the support of the head given not the body. For each single branch we use the method explained earlier to calculate the support pairs and the same stopping criterion. After building a decision tree using the algorithm described above we update the subdatabase by subtracting the explained part of the probability for that target concept. We repeat building branches till all examples for are explained, i.e. the remaining probability is zero for all examples in the subdatabase.

It is possible that one target concept is explained using some background concepts which are not necessary to explain the other target concepts. In fact, the branches for the different target concepts can contain different background concepts in different combinations. We hope to achieve considerable fewer rules than MA-ID3, Baldwin (1998), which builds one decision tree for all words on the target relation where the branches are the same for all words and just the supports at the end of the branches differ. In this case the same branches and background concepts are needed in order to explain all target concepts. Because in our case the target relation is split up in single concepts which are treated separately, we need fewer branches and background concepts in order to explain the single target concepts. All resulting branches for a specific word on the target relation are translated into Fril rules where these rules are connected in an extended Fril rule, Baldwin (1995). For classification problems the label of the target relation whose extended Fril rule provides the highest calculated probability is the resulting classification for a new data point. When dealing with prediction problems we defuzzify the calculated probabilities in order to receive the predicted value for a new data point.

In the future we also try to involve the negation of words hoping that this results in fewer rules.

After we generated a decision tree, i.e. learned the rules for a word on the target relation, we want to add these rules to the background knowledge, such that they can be used to explain the other words on the same target relation or other target relations. The constructed rules are then dependent on the order we built the decision trees.

DISCUSSION

The algorithm presented here for inducing probabilistic rules, which generalise probabilistic databases, provides some advantages over classical ILP methods. Like in classical ILP we infer rules that are easily understood by people, and also we are able to express relations between objects instead of just propositions. In the same way as ILP we can therefore describe concepts which cannot be easily described in an attribute-value language.
However, in addition to classical ILP which is limited to the normal bivalent logic without involving any uncertain concepts, we can model probabilistic problems. If this method is combined with the use of fuzzy partitions on the attribute universes we can also express vague concepts.
Some ILP algorithms can find recursive definitions for their target relations or even invent new predicates in order to explain the examples. A future goal is to improve our method and try to reach these capabilities as well.

REFERENCES

Baldwin, Jim F.; Martin, Trevor P.; Pilsworth, Bruce W., 1988, “Fril Manual (version 4.0)”, FRIL Systems Ltd, Bristol Business Centre, Maggs House, Queens Road, Bristol BS8 1QX, UK.

Baldwin, Jim F.,1990, “Computational Models of Uncertain Reasoning in Expert Systems”, Computers and Mathematics with Applications, Vol. 19, No. 11, pp. 105-119.

Baldwin, Jim F.; Martin, Trevor P.; Pilsworth, Bruce W., 1995, “Fril – Fuzzy and Evidential Reasoning in Artificial Intelligence”, Research Studies Press Ltd.

Baldwin, Jim F.; Lawry, Jonathan; Martin, Trevor P., 1996, “A Mass Assignment Theory of the Probability of Fuzzy Events”, Fuzzy Sets and Systems, Vol 83, pp. 353-367.

Baldwin, Jim F.; Lawry, Jonathan; Martin, Trevor P., 1998, “Mass Assignment Based Induction of Decision Trees on Words”, Proceedings IPMU 1998, Paris, France, pp. 524-531.

Jeffrey, Richard C., 1965, “The Logic of Decision”, Gordon & Breach Inc., New York.

Muggleton, Stephen, Buntine Wray, 1992, “Machine Invention of First-Order Predicates by Inverting Resolution”, Inductive Logic Programming, ed. by St. Muggleton, Academic Press Limited, pp. 261-280.

Muggleton, Stephen; De Raedt, Luc, 1994, “Inductive Logic Programming: Theory and Methods”, The Journal of Logic Programming, Vol. 19/20, pp. 629-679.

Quinlan, John R., 1990, “Learning Logical Definitions from Relations”, Machine Learning, Vol. 5, No. 3, pp. 239-266.

Quinlan, John R.; Cameron-Jones, Richard M., 1995, “Induction of Logic Programs: FOIL and Related Systems”, New Generation Computing, Special Issue on ILP, Vol. 13, pp. 287-312.

Zadeh, Lotfi A., 1975/76, “The Concept of a Linguistic Variable and its Applications to Approximate Reasoning”, Part I: Information Sciences 8, pp. 199-249, Part II: Information Sciences 8, pp. 301-357, Part III: Information Sciences 9, pp. 43-80.