Fuzzy FOIL: A Fuzzy Logic Based Inductive Logic Programming System

Guiming Chen

Thesis submitted to
the School of Graduate Studies and Research
in partial fulfillment of the requirements
for the Master degree in Computer Science

The Ottawa-Carleton Institute for Computer Science
University of Ottawa

ISBN 0-612-15707-5

UNIVERSITÉ D'OTTAWA
UNIVERSITY OF OTTAWA

Acknowledgments

I would like to thank all the people who assisted me in completing this thesis work.

First, I would like to express my deepest appreciation to my supervisor Prof. Stan Matwin for his insightful guidance, constructive suggestions, and ideas throughout my graduate studies. His accuracy in reviewing and modifying the drafts of the thesis has greatly improved the contents and its presentation.

I am also much indebted to Dr. Bob Orchard for his assistance in using Fuzzy Shower Control Simulator to collect training data, which was essential in the experimental evaluation of Fuzzy FOIL.

Finally, I would like to acknowledge the partially financial support received from NSERC for the thesis research work.

Table of Contents

ABSTRACT

In many domains, characterizations of a given attribute are imprecise, uncertain and incomplete in the available learning examples. The definitions of classes may be vague. Learning systems are frequently forced to deal with such uncertainty. Traditional learning systems are designed to work in the domains where imprecision and uncertainty in the data are absent. Those learning systems are limited because of their impossibility to cope with uncertainty -- a typical feature of real-world data.

In this thesis, we developed a fuzzy learning system which combines inductive learning with a fuzzy approach to solve problems arising in learning tasks in the domains affected by uncertainty and vagueness. Based on Fuzzy Logic, rather than pure First Order Logic used in FOIL, this system extends FOIL with learning fuzzy logic relation from both imprecise examples and background knowledge represented by Fuzzy Prolog. The classification into the positive and negative examples is allowed to be a degree (of positiveness or negativeness) between 0 and 1. The values of a given attribute in examples need not to be the same type. Symbolic and continuous data can exist in the same attribute, allowing for fuzzy unification (inexact matching). An inductive learning problem is formulated as to find a fuzzy logic relation with a degree of truth, in which a fuzzy gain calculation method is used to guide heuristic search. The Fuzzy FOIL's ability of learning the required fuzzy logic relations and dealing with vague data enhances FOIL's usefulness.

1. Introduction

Inductive Logic Programming employs techniques from both machine learning and logic programming. By using computational logic as the representational mechanism, inductive learning acquires an elegant first order logic based formalism to represent hypotheses and background knowledge. By using inductive rather than deductive reasoning as the mechanism of inference, inductive logic programming also extends the theory and practice of computational logic. This makes inductive logic programming more powerful than traditional learning techniques.

Therefore, Inductive Logic Programming is currently regarded as one of the interesting and new developments in the Machine Learning community. Some empirical inductive logic programming systems have already shown their potential for practical applications. Significant effort has been devoted to improve the usefulness and robustness of ILP in real life domains.

Empirical Inductive Logic Programming systems usually learn from a large collection of examples. The larger size of the example set, the higher rate of various kinds imperfect data exist. The available training data and background knowledge are imprecise, uncertain and incomplete in real life domains. Most Inductive Logic Programming systems seem to only work in ideal domains or lack efficient techniques to handle imperfect data. Machine learning systems are often faced with the imperfection of data. It is agreed that the key to successfully apply inductive logic programming system in practical application is the ability to deal with imperfect data [Lavrac, N. and Dzeroski, S. 1994].

Recently, the research on improving learning system's ability to handle imperfect data is receiving a growing interest. Several systems have included variety of techniques for handing imperfect data. Until recently, most systems have provided mechanisms for handling random and noisy data. Such mechanisms are often called noise-handling mechanisms. Most of the techniques of those mechanisms are similar to pruning (pre-pruning and post- pruning) of decision trees and use significance tests in rule truncation.

They are used in C4.5 [Quinlan 1993], CN2 [Clark and Niblett 1989, Clark and Boswell 1991]. Another noise-handling mechanism used in tree pruning [Cesnik and Bratko 1991] and rule truncation [Clark and Boswell 1991] is improved probability estimates [Cestnik 1990].

During the early stage of research in Inductive Logic Programming, most systems were interactive and incremental in nature. The examples had been assumed correct, as well as the answers provided by the oracle. Therefore, the issue of data imperfection was not a major concern for an ILP researcher. However, recently, with significant progress of research on empirical ILP systems, ILP systems have reached the stage of application in practical domains where real data is not so ideally correct and size of example space is increasing. In order to improve the efficiency of ILP systems in real domains, the interest in handling imperfect data in ILP has increased. FOIL [Quinlan & Cameron-Jones 1993] and LINUS [S. Dzeroski & N.Lavrac 1991] include sophisticated noise-handling mechanisms. Recent mFOIL[Dzeroski 1991], an extension of FOIL, uses search heuristics and stopping criteria to improve noise-handling. A post-processing mechanism based on reduced error pruning has also been used in FOCL [Pazzani et al. 1991, Brunk and Pazzani 1991], a system that extends FOIL by combining empirical and explanation-based learning. ILP systems based on generalization techniques (Bottom-up), such as GOLEM [Muggleton and Feng 1990] and CIGOL [Muggleton and Buntine 1988], have been extended to include a measure of information compression based on the noise-handling approach proposed in [Muggleton, S., Srinivasa, A., and Bain, M. 1992]. The noise-handing mechanisms used in the systems mentioned above have been successfully applied to a wide range of real life domains.

However, statistical noise is not the only source of imperfect data problems. When learning the definitions of logic relation, there are usually four types of imperfect data encountered in real life domains [Lavrac and Dzeroski 1992]:

- noise, i.e., random errors in the training examples and background knowledge

- insufficiently covered example space. i.e., too sparse training examples from which it is difficult to detect correlation.
- inexactness, i.e., inappropriate or insufficient description language which does not contain an exact description of the target concept.
- missing values in the training examples.

Current ILP systems usually incorporate a single noise-handing mechanism to handle the first three types of imperfect data. The problem here is that the first type of imperfect data -- noise, and the third type of data -- inexactness, represent two different types of uncertainty, i.e. randomness and vagueness. They are inherently different. Randomness is a type of probability uncertainty concerned with the uncertain relationship between cause and effect, while vagueness is a type of possibility uncertainty concerned with an object's uncrisp boundary and vague character. There is little in common between noise and vagueness.

The limitation of current mechanisms for handling imperfect data is that they use probability to deal with possibility uncertainty. The kinds of possibility uncertain data, such as vague data and imprecise data, either are treated as noise, random error or are absent in the training examples. There is no proper mechanism to distinguish probability uncertainty and possibility uncertainty in current ILP.

One of the reasons that possibility uncertainty is mistreated as noise or as randomness error is that the difference between possibility and probability uncertainty are barely noticed in Machine Learning research area. Another reason is that there have been some arguments about the relation and difference between probability uncertainty and possibility uncertainty[Elkan, 1993.]. Some researches in probability and statistics even claim that uncertainty is the same as randomness, and probability theory is good enough for handling uncertainty. Fortunately, the development of research on fuzzy theory and its application have shown the advantages of using fuzzy theory and fuzzy logic to deal with imprecision and vagueness - the kinds of possibility uncertainty in real life. At the moment, it is agreed

that there exist two main kinds of uncertainty, i.e. randomness and fuzziness, as far as uncertain data are concerned. To make the difference between two kinds of uncertainty more clear, the difference will be explained and discussed in detail in the next chapter .

In real domains, there exists a great deal of uncertain or vague data, which can never be formulated in a certain and well-defined form. In many cases, information is naturally imprecise or fuzzy, e.g. when representing personalities, physical features of individuals, and subjective opinions and judgments in situations such as medical decision making, economic forecasting, and personal evaluation. So, the fuzzy information reflects an important aspect of concepts in learning systems. If fuzzy data were treated as absent, it would certainly impact negatively the accuracy of learned concept.

To overcome the lack of mechanism to handle fuzzy data in current ILP systems, a fuzzy Inductive Logic Programming system - Fuzzy FOIL has been developed in this thesis. The main contribution of Fuzzy FOIL is that it extends the ability of FOIL with learning fuzzy logic relation from both imprecise examples and background knowledge represented by Fuzzy Prolog. Fuzzy membership for describing the possibility distribution of fuzzy sets can be automatically built by using a simple linear regression technique. A new definition of completeness and consistency under the fuzzy theory framework is proposed to ensure that the learned relations cover all positive examples and no negative examples, and are consistent with the background knowledge. The system with its theoretical basis and practical implementation, can handle imperfect data better than FOIL.

The thesis is organized as follows. Chapter 2 briefly describes the basic principle of fuzzy theory, and discusses the difference between two types of uncertainty, possibility and probability. Chapter 3 starts with a short survey of related research and then describes the hypothesis language and learning bias in Fuzzy FOIL. Chapter 4 gives new definitions of notion of fuzzy coverage and fuzzy θ-subsumption in the fuzzy framework. Chapter 5 introduces a fuzzy top-down specialization algorithm and a fuzzy gain heuristic to guide the search in Fuzzy FOIL. Chapter 6 describes a method to automatically construct a

membership function. Chapter 7 presents experimental results of Fuzzy FOIL's performance on three examples: learning *'strong_man'* relation, learning *'close-friend'* relation, and learning *'fuzzy shower control rules'*. In Chapter 8, the limitations of the system and future work are discussed. Finally, general conclusions are presented.

2. Uncertainty, Fuzziness and Fuzzy Set Theory

In this thesis, we are concerned with handling uncertain data in an inductive logic learning system. In real life, people are familiar with randomness, and are used to apply probability to deal with uncertainty. A new concept – fuzziness [Zadeh, L.A. 1965] – has been suggested as an alternative to randomness for describing uncertainty. We are often faced with the question: is uncertainty the same as randomness? If we are not sure about something, is it only up to chance? Some people answer to the question with 'yes' and claim that the only satisfactory description of uncertainty is probability, and they raised many questions about the adequacy and practical usefulness of fuzziness [Elkan, 1993]. But other people believe that fuzzy theory and fuzzy logic can do better than probability theory in real life domains, and that only fuzziness is adequate to handle many situations involving uncertainty [Klir 1989] [Kosko, 1990].

Our opinion here is clear: experience and experiment tell us that classical probability theory is insufficiently expressive to cope with the multiplicity of kinds of uncertainty in the tasks of learning system from imperfect data. To make the problem more specific, here we only focus on discussing the difference between two kinds of uncertainty, i.e., possibility and probability, and ignore other debates in fuzzy community and probability community. The limitation of this discussion is justified because, in learning, the most important issue is that vague data is mistreated as random noise, or is omitted from the training data. This limits the learning systems to "ideal" domains. Moreover, the learned relations and concepts are incomplete, because they do not cover some important aspects of information implied by vague and imprecise data. Our discussions begin with introducing the concept of fuzziness.

2.1 Fuzziness

We are presented with fuzzy information everyday. The natural language we use is essentially fuzzy. For example,

- *Big* trucks must go *slowly*.

- *Strong* man is *more suitable* for doing *heavier* work.
- *Smart* student often gets *high* grade
- Smith is *very young* and *tall*, and his intelligence is *about average*.

Fuzziness occurs when the boundary of a piece of information is not clear-cut. For instance, the concepts used in above examples, *tall, young, high, smart, big, slowly* are fuzzy. There are no single quantitative value and well-defined formula to define the term *young*. For some people, age 20 is young, and for others age 35 is somewhat young. Obviously, concept *young* has no clear boundary. Age 1 is definitely young and age 100 is definitely not young, however, age 35 has some possibility of being young. The possibility of fuzzy concept usually depends on the context in which it is being considered. The concept *tall* is a simple example: people with height of 175 cm may be considered as tall in Japan, but may not be considered tall in United States.

2.2 The Difference Between Probability and Possibility Uncertainty

In general, the primary distinction between the two different uncertainties -- probability and possibility -- can be described as follows:

- Probability theory, which is based on classical binary logic measures the occurrence of an event E in a measurable subset of the sample space. An event E either occurs or does not occur, it cannot occur to a degree. Possibility theory is based on fuzzy logic, namely, an element can belong to a fuzzy set with a degree of membership.
- In applications, probabilities are usually objective physical constants. A frequent interpretation of probability is used to estimate the frequency of the outcome of a random event. However, the possibility of class membership comes from the definition of the class and is not directly related to any notion of "frequency of occurrence". Possibilities are used to measure personal or subjective belief in a fuzzy set or proposition.

- Probability corresponds to "What may happen" and "How likely is it that X may happen"; possibility corresponds to "What can be done" and "How likely is it that X is doable"?
- Probability theory is used to observe random phenomena; possibility theory is a tool for studying fuzziness. More specifically, probability theory does not provide a general computational system for representing the meaning of fuzzy propositions containing fuzzy predicates, fuzzy quantifiers, and fuzzy probabilities [Zadeh, 1984], and does not provide a general computational system for inference from fuzzy propositions, whereas possibility theory does.

On the other hand, the two different kinds of uncertainty – probability and possibility have some things in common:

- Both probability and possibility are concerned with some type of uncertainty and both use the [0,1] interval for their measurements of the range of their respective functions.
- Probability and possibility are both relative to a certain frame of reference directly or indirectly.
- The intuitive interpretation is that possibility is a quantitative mathematical concept analogous to probability. A high degree of probability always implies a high degree of possibility, but not the converse, i.e. a high degree of possibility does not imply a high degree of probability, neither does a low degree of probability imply a low degree of possibility. However, if an event is impossible, it is bound to be improbable.

To end the discuss and to illustrate the difference between probability and possibility more specifically, an example [Li, Deyi and Liu, Dongbo 1990] is presented below.

Consider the statement "Hans ate X eggs for breakfast," with X taking values in $U = \{1,2,3,...\}$. We may associate a possibility distribution with X by interpreting $\mu(x)$ as the degree of ease with which Hans eats x eggs. We may also associate probability distribution with X by interpreting $p(x)$ as the probability of Hans eating x eggs for breakfast. Assuming that we employ some explicit criterion for assessing degree of ease with which

Hans can eat X eggs for breakfast, the values of $\mu(x)$ and $p(x)$ might be as shown in Table 2.2.

"Hans ate X eggs for breakfast"

X	1	2	3	4	5	6	7
$\mu(x)$	1	1	1	1	0.8	0.6	0.4
$p(x)$	0.1	0.8	0.1	0	0	0	0

$\mu(x)$: Possibility of "Hans ate X eggs for breakfast"

$p(x)$: Probability of "Hans ate X eggs for breakfast"

Table 2.2

We observe that whereas the possibility that Hans may eat 3 eggs for breakfasts is 1, the probability that he may do so might be quite small, e.g. 0.1. Thus, a high degree of possibility does not imply a high degree of probability, nor does a low degree of probability imply a low degree of possibility. However, an impossible event must be also improbable.

2.3 Fuzzy Set Theory

Since the invention by Lofti Zadeh in the mid 1960s, the fuzzy theory approach has clearly captured the interest of many researchers around the world and has been used to develop many applications.

The representation of this type uncertainty information in Fuzzy FOIL is based on Fuzzy Set Theory[Zadeh, 1965].

Definition 2.1 Fuzzy Set:

Let U be a collection of objects which report the universe of discourse. A fuzzy set F in U is characterized by a membership function of : $U \in [0,1]$, with $\mu_F(u)$ representing the grade (possibility) of membership of $u \in U$ in the fuzzy set F. A fuzzy set may be viewed

as a generalization of the concept of an ordinary set (a crisp set) whose membership function only takes two values {0, 1}.

For example, Figure 2.1 shows the membership functions of three fuzzy sets, namely, 'slow', 'medium', and 'fast' for the speed of a car. In this example, the universe of discourse is all possible speeds of the car; that is, $U = [0, V_{max}]$, where V_{max} is the maximum speed of the car. At a speed of 40 mph, for example, the fuzzy set "slow" has membership value 0.7, that is $\mu_{slow}(40) = 0.7$, the fuzzy set "medium" has membership value 0.7, that is $\mu_{medium}(40) = 0.7$, and the fuzzy set "fast" has membership value 0, that is, $\mu_{fast}(40) = 0$.

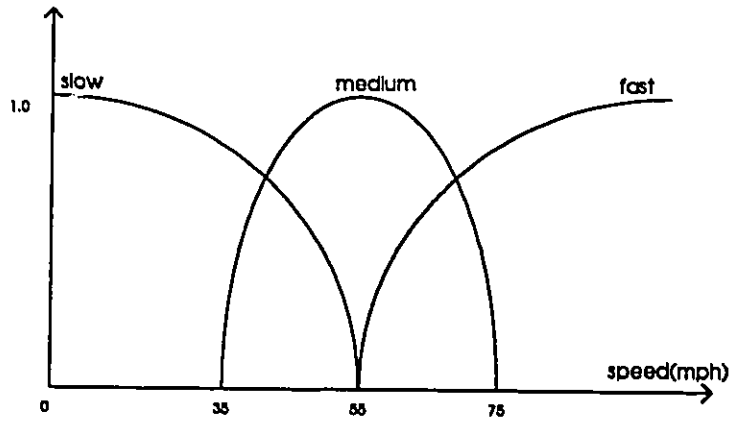


Figure 2.1 Membership of three fuzzy sets: 'slow', 'medium', and 'fast'

Definition 2.2. Support, Center, and Fuzzy singleton:

The support of a fuzzy set F is the crisp set of all points $u \in U$ such that $\mu_F(u) > 0$. The center of a fuzzy set F is the point(s) $u \in U$ at which $\mu_F(u)$ achieves its maximum value. If the support of a fuzzy set F is a single point in U at which $\mu_F(u) = 1$, the F is called a fuzzy singleton representation.

For example, the support of fuzzy set 'slow speed' should be a set of value in the interval [0, 55], i.e. $F_{slow} = \{u | u >= 0, u <= 55\}$. And its center point is at 0, where $\mu_F(0)$ achieves its maximum value 1.0.

If fuzzy set 'slow' , 'medium', and 'fast' are represented by the points 0, 55, and 90 respectively, these fuzzy sets are called singleton fuzzy model.

Definition 2.3. Intersection, Union, and Complement:

Let A and B be two fuzzy sets in U. The intersection $A \cap B$ of A and B is a fuzzy set in U with membership function defined for all $u \in U$ by

$$\mu_{A \cap B}(u) = min\{\mu_A(u), \mu_B(u)\} \quad (2.1)$$

The union of $A \cup B$ of A and B is a fuzzy set in U with the membership defined for all $u \in U$ by

$$\mu_{A \cup B}(u) = max\{\mu_A(u), \mu_B(u)\} \quad (2.2)$$

The complement $\sim A$ of A is a fuzzy set in U with the membership function defined for all $u \in U$ by

$$\mu_{\sim A}(u) = 1 - \mu_A(u) \quad (2.3)$$

For instance, at the speed of 50 km point,

$$\mu_{slow}(50) \cap_{medium} (50) = min\{\mu_{slow}(50), \mu_{medium}(50)\} = \mu_{slow}(50)$$

$$\mu_{slow}(50) \cup_{medium} (50) = max\{\mu_{slow}(50), \mu_{medium}(50)\} = \mu_{medium}(50)$$

And at speed of 75 km,

$$\mu_{\sim medium}(75) = 1 - \mu_{medium}(75) = 1.0 - 0.0 = 1.0.$$

Definition 2.3 shows only one possible choice of operators for intersection, union, and complement. One choice of operator corresponds to one interpretation of the meaning of the logic operation's intersection, union, and complement. Based on different interpretations which range from intuitive argumentation to empirical axiomatic justifications, other operators have been suggested in the literature.

Definition 2.4 The Extension Principle

Let U and V be two universes of discourse and f be a mapping from U to V. For a fuzzy set A in U, the extension principle defines a fuzzy set B in V by

$$\mu_B(v) = \text{Sup}_{u \in f^{-1}(v)} (\mu_A(u)) \qquad (2.4)$$

That is $\mu_B(v)$ is the superium of $\mu_A(u)$ for all $u \in U$ such that $f(u) = v$, where $v \in V$ and we assume that $f^{-1}(v)$ is not empty. If $f^{-1}(v)$ is empty for some $v \in V$, define $\mu_B(v) = 0$. The extension principle is a tool for generalizing crisp mathematical concepts to fuzzy sets. It has been extensively used in the fuzzy literature.

For example, let $U = \{0,1,2,..,9\}$. V is the set of natural numbers. f is a mapping from U to V,

$$f: x \rightarrow 2x$$

A fuzzy set A in U is defined by

$$A = \{1/0, 1/1, 0.7/2, 0.5/3, 0.2/4\}$$

where 1/1 and 0.7/2 mean that the degree of number 1 being a member of fuzzy set A is 1 and the degree of number 2 being a member of A is 0.7.

By applying the definition , we obtain

$$B = f(A) = \{1/0, 1/2, 0.7/4, 0.5/6, 0.2/8\}$$

Definition 2.5 Linguistic Variables and Hedges

There are two interpretations for the concept of a linguistic variable. Formally, a linguistic variable is defined as follows [Zadeh, 1975]:

Definition Linguistic Variables (formal):

A linguistic variable is characterized by a quintuple $(x, T(x), U, G, M)$ in which x is the name of variable; $T(x)$ is the term set of x, that is, the set of names of linguistic values of x with each value being a fuzzy set defined on U; G is a syntactic rule for generating the names of values of x; and M is a semantic rule for associating each value with its meaning.

This definition may give the reader a feeling that the linguistic variable is a complex concept, but in fact it should not be. The goal of introducing the concept of a linguistic variable is to present a formal way of saying that a variable may take words in natural

languages as its values. For example, if we can say "the speed is fast", then the variable speed should be understood as a linguistic variable; but this does not mean that the variable speed cannot take real values. In this spirit, we have the following intuitive definition of a linguistic variable.

Definition Linguistic Variables (intuitive):

If a variable can take words in natural languages (for example, "small", "fast", and so on) as its values, this variable is defined as a linguistic variable. These words are usually labels of fuzzy sets. A linguistic variable can take either words or numbers as its values.

For example, the linguistic variable speed can take "slow", "medium", and "fast", as its values (Fig. 2.1). It can also take any real numbers in the interval [0, Vmax] as its values. Linguistic variable is an important concept that gives us a formal way to quantify linguistic descriptions about variables.

Since in linguistic descriptions we often use hedges, such as "very", "more" or "less", to describe other terms, we need formal definitions for what these hedges mean. Although in everyday use the hedge "very" does not have a well-defined meaning, in essence it acts as an intensifier. In this spirit, we have the following definitions for the two most commonly used hedges: "very" and "more or less".

Definition Hedges:

Let F be a fuzzy set in U (for example, F = small), then "very F" is defined as a fuzzy set in U with the membership function

$$\mu_{\text{very F}}(u) = (\mu_F(u))^2 \qquad (2.6)$$

and "more or less" is a fuzzy set in U with membership function

$$\mu_{\text{more or less F}}(u) = (\mu_F(u))^{1/2} \qquad (2.7)$$

where $u \in U$. For example,

F = small, $\mu_{\text{small}}(u) = \{1.0/1 , 0.6/2 , 0.1/3 , 0.0/4\}$.

F = very small, $\mu_{\text{very small}}(u) = (\mu_{\text{small}}(u))^2 = \{1.0/1, 0.36/2 , 0.01/3, 0.0/4\}$.

3. Fuzzy FOIL : A Fuzzy Logic Based Inductive Logic Programming System

3.1 The basic idea

There are two fundamental techniques used in the empirical inductive learning programming system : generalization technique (bottom-up induction) and specialization technique (top-down induction). FOIL (Quinlan, 1990) is an efficient top-down first-order inductive learning system for inducing function-free horn clause logic relations. FOIL's outer loop algorithm is a greedy covering procedure that learns one clause at a time. Each clause is constructed to maximize coverage of positive examples while excluding all negatives.

Fuzzy FOIL is a prototype extension system of FOIL. It is designed for incorporating fuzzy logic in a specialization technique to complement the two major limitations of FOIL -- inability to deal with uncertainty information and insufficient strength in coping with continuous number features (See example 2 in chapter 7). Fuzzy FOIL is also a more realistic inductive learning system in terms of the ability of handling imperfect data in the real world.

Based on fuzzy logic, Fuzzy FOIL learns fuzzy relations from examples and background knowledge which are imprecise and vague. The facts in examples and rules in background knowledge are represented in Fuzzy Prolog -- a representation language more powerful than the standard Prolog. The classifications into positive and negative examples are allowed to be a degree of positiveness and negativeness. The learned relations also can be true to a certain degree, which depends on the evidence of examples. In theory, when the degrees are limited to be either value 0 or 1, FOIL is a special case of fuzzy FOIL in which data and background knowledge involve no possibility uncertainty. To make sure the learned relations are complete and consistent, a notion of fuzzy coverage is introduced. A fuzzy θ-subsumption is defined for structuring the hypothesis space. Based on the definition of fuzzy coverage and fuzzy θ-subsumption, a fuzzy unifying framework

for specialization is designed to support top-down search of refinement algorithm. The system was implemented using Quintus-Prolog 3.1 .

In the following, we first briefly survey some related researches. Then give the description of the system by introducing the hypothesis language and language bias used in Fuzzy FOIL.

3.2 A Survey of Related Research

Inductive learning from vague and imprecise data has not been studied in much detail. There are no published experimental results, or standard benchmark datasets, directly concerning this topic. Only one article [Yufei Y, Michael J.S, 1995] on learning fuzzy decision trees presents a result from a simple experiment.

There are two distinct research directions in inductive learning in imprecise and uncertain environments. One is the research on fuzzy ILP (inductive logic programming). Until recently, there were no direct results on fuzzy ILP. Our work establishes the first experimental research result on Fuzzy ILP. Another direction is the research on fuzzy decision trees. Yufei's Fuzzy Decision Trees are the latest published work in this area. In this work, a fuzzy decision tree induction method, which is based on the reduction of classification ambiguity with fuzzy evidence, is presented. Each fuzzy evidence is the knowledge about a particular attribute. The learning method is similar to Quinlan's ID3 decision tree induction method [Quinlan 1986].

There are several differences between our approach and fuzzy decision tree approach.

First, the major difference between the two approaches is the use of hypothesis representation. Fuzzy FOIL is based on Inductive Logic formalism. Fuzzy decision tree is based on Quinlan's ID3 propositional attribute-value formalism. The propositional attribute-value formalism is simple but limited. Fuzzy FOIL not only takes the advantages of better representation power of first-order logic formalism, but also extends the

powerful representation of pure logic further with the ability to handle uncertain data, which is critical for improving the efficiency of ILP in real domains.

Second, in contrast to the approach of using truth level threshold in classification in learning fuzzy decision tree, two important learning principles – completeness and consistency – are defined in the context of imprecise and uncertain data for Fuzzy FOIL. These two principles guarantee that learned relations cover positive imprecise examples completely, and that no negative examples are covered. Therefore Fuzzy FOIL stands on a good theoretical basis.

Third, when the given data is numeric, the data need to be fuzzified into symbolic linguistic terms before learning takes place in Yufei's system. Fuzzy FOIL uses 'fuzzy unification' technique to perform inexact matching between numerical data and linguistic terms, which is a more flexible approach.

3.3 Hypothesis Language Used in Fuzzy FOIL

The main contribution of FOIL is to recognize the power of logic programming as a representation language for inductive learning. However, although pure logic programming language Prolog, which is based on the first order logic, has advantages over the decision tree approach in the ability of representing knowledge, it is still not powerful enough to represent fuzzy concept and fuzzy relations in real life. A serious shortcoming of pure first order logic is its inability to cope with the issue of uncertainty and imprecision. As a consequence, conventional Prolog does not provide an appropriate conceptual framework for dealing with uncertainty information. It can only deal with "ideal" data and knowledge which must be represented exactly. Most processing in Prolog assumes that the information represented is exact, correct and well-formulated. The answer to a Prolog query must definitely be either true or false. To illustrate the limitation of pure Prolog, let's see the following examples:

Suppose there is a Prolog predicate tall(Name) defined as :

```
tall(Name):-
    candidate(Name, Height, _Age),
    Height >= 180.
```

This predicate defines that a person can be considered as tall person only if his height is equal or greater than 180 cm. Now, suppose we have the following listed candidates in Prolog database:

```
candidate(tom,190).
candidate(smith,179.7).
candidate(susan,about185).
candidate(mary,165).
```

By the query ?-tall(Name), only one candidate meets the definition, e.g.,

```
?-tall(Name).
Name = tom;
no.
```

Among four candidates, only Tom is considered a tall person. We can notice that for candidate Smith, in spite of his height 179.7, which is almost equal to 180cm, he still can not be considered as a tall person. For another candidate Susan, whose height is about 185cm which is definitely greater than 180cm, unfortunately, Prolog was not able to recognize that. It is bound to fail when matching term 'about185' with numerical value 180 in pure Prolog.

This example shows that the results deduced by conventional Prolog are not consistent enough with the facts in real world.

Obviously, the above limitation of representing imprecise and uncertain data impacts applications of the traditional pure logic-based ILP systems in real domains. To overcome the limitation of traditional ILP system, Fuzzy-Prolog [Chen 1993, Li, 1989] is used in our system. By means of Fuzzy Prolog, fuzzy data and incomplete data can be handled

appropriately. For the above example, the answer provided by Fuzzy-Prolog system would be :

?-{Possibility}:tall(Name).

Name = tom

Possibility = definite;

Name = smith

Possibility = almost_definite;

Name = susan

Possibility = definite;

Name = mary

Possibility = impossible;

no.

Seemingly, this result is more flexible than the result deduced from conventional Prolog and is closer to real life.

The reason that Prolog fails to give satisfactory answers in this example is that it lacks built-in mechanisms to handle imprecise, vague information. In this example, the predicate tall is a fuzzy concept. The value of Smith's and Susan's height 179.7 and 'about185' are also inexact and vague. They represent physical features of individuals which are found naturally to be imprecise or uncertain in the real world. One way to let Prolog deal with imperfect data is to pre-process the data so that all kinds of data are "ideal" enough for Prolog to process, e.g., change value 179.7 to 180 and 'about185' to 185. This approach is often used in current Prolog-based AI systems, especially, in current Inductive Logic Programming Systems which usually do a 'pre-processing of examples' [Lavrac and Dzeroski 1994, p58] before applying the systems to learn a relation from examples. This extra process not only limits application of ILP in 'ideal' domains rather than real life domains, but also loses important information represented by fuzzy concept. To overcome

the disadvantage of pure Prolog, in our system, we move to a more powerful representation language - Fuzzy Prolog.

3.3.1 A More Powerful Representative Language - Fuzzy Prolog

The hypothesis language used for representing expected learning relation in fuzzy FOIL is function-free fuzzy horn clause defined in Fuzzy-Prolog. Based on fuzzy logic rather than bivalent logic, Fuzzy Prolog, as its name suggests, is the extension of pure Prolog.

In order to have a complete view of Fuzzy-Prolog, we briefly describe the basic concept of Fuzzy Prolog below.

3.3.1.1 Fuzzy First Order Horn clause

Prolog is formulated based on first-order Horn Clause, Fuzzy-Prolog is defined based on fuzzy first-order Horn clause. Fuzzy Horn clause is established by associating each Horn clause with a numerical implication strength f, or called truth value f. Conventional variables and terms defined in pure Prolog are extended as Fuzzy Linguistic variables and fuzzy terms in Fuzzy Prolog. We first introduce the notion of fuzzy first order horn clause. The concept of Fuzzy Linguistic Variable and Linguistic value have been introduced in section 2.5 and will be discussed in detail in the section 3.5.

Based on Horn clauses, there are also two kinds of fuzzy-Horn clauses:

The first case, a fuzzy-Horn clause consists of one conclusion and n (≥ 0) conditions

(1) $\quad A \leftarrow \{f\}: B_1 \wedge B_2 \wedge \ldots \wedge Bn.$

A special case of (1) is a f-fact, which consists of only one assertion with a truth degree f, i.e.

$$A \leftarrow \{f\}.$$

The second case, a fuzzy clause consists of no conclusions but n (≥ 0) conditions:

(2) $\quad \leftarrow \{f\}: B_1 \wedge B_2 \wedge \ldots \wedge Bn.$

This kind of fuzzy horn clause is interpreted as a fuzzy query goal.

Based on the definition of fuzzy horn clause above, a program in Fuzzy-Prolog mainly consists of three parts:

(1) *f*-Assertion:

P(t1,t2,...,tn):-{*f*}

This means that there exists a relationship (fact) P, among terms t1,t2,...,tn, with the degree of belief *f*.

(2) *f*-Rule:

P(t1,t2,...,tn):-{*f*}:Q1,Q2,...,Qn.

$f_p = f \times \min\{tv(Qi) | i=1,2,...,n\}$;

This is a fuzzy rule which defines an **IF-THEN** rule with implication strength of *f*. By applying minimal principle defined in definition 2.3, the truth degree of a certain instance of rule is measured as $f \times \min\{tv(Qi)\}$, where tv(Qi) is the truth value of Qi, i= 1,2,..,n.

(3) f-Goal:

?-{*f*}:Q1,Q2,...,Qn. (*f* is a constant)

This is a fuzzy query clause, which has two cases and meanings: in the case when *f* is a constant, a fuzzy query means whether the n sub-goals Q1,...,Qn can all be satisfied over the threshold *f*.

?-{*F*}:Q1,Q2,...,Qn. (*F* is a variable)

When implication strength is a variable, a query is a question for answering the truth value *F* of conjunction of predicates: Q1,...,Qn.

3.3.1.2 Fuzzy Inference (Approximate Reasoning) in Fuzzy Prolog

A fuzzy procedure is interpreted as a fuzzy rule of question answering. For a given question ?- {f1}:A1 ^ A2 ... ^An., there are two cases:

- **The first case:** f1 is a constant:

?-{f1}:A1 ^ A2 ... ^An, (n >= 1)

it corresponds to the n sub-goals below:

?-{f1}:A1,

?- {f1}:A2,

• • •

?- {f1}:An.

We may select f-Procedure A1 to prove the subquestion:

?-{f1}:A1.

Now suppose f-Procedure

A ←{f2}:B1 ^ B2 ^ ... ^ Bm, m >=0

exists in database. If A1 matches A, f1 = < f2 and the most general substitution exists between A1 and A, the new f-sub goals are derived as:

?- {f1'}:(B1 ^ B2 ^ ... ^ Bn)

?-{f1 }: A2,

• • •

?-{f1 }:An.

where f1' = f1/f2.

The derivation method is reiterated for the present f-Goal, until the computation is terminated by the derivation of the halt statement. The set of all f-Horn clause rules can be seen as a nondeterministic programming language, because, given a single f-Goal, several f-Procedures may have a name which matches the selected f-Goal at the same time. Each f-Procedure gives rise to new f-Goal.

- **The second case: f1 is a variable:**

In this case, f1 is a variable which represents an unbounded truth value of Goal. It means that you need to deduce f1. In the proof procedure, each value of f is given according to the min-max rule for an **AND/OR** tree.

For the **AND** tree:

A ← {f}:B1 ^ B2 ^ ... ^ Bn

For goal: $? \leftarrow \{F\}:A.$

the truth value of F should be :

$$F = f \times \min\{tv(Bi) | i=1,2,...,n\}$$

where tv(Bi) is the truth value of each subgoal Bi.

For the **OR** tree:

$$A \leftarrow \{f1\}:B11 \wedge B12 \wedge ... B1I1$$

$$A \leftarrow \{f2\}:B21 \wedge B22 \wedge ... B2I2$$

$$A \leftarrow \{fn\}:Bn1 \wedge Bn2 \wedge ... BnIn$$

for the query $? \leftarrow \{F\}:A.$

the truth value of F should be:

$$F = \max\{fi \times \min \{ tv(Bij) | i=1,..,n; j = 1, ..., Ii \}\}.$$

3.4 Training Example and Background Knowledge

3.4.1 Representing Fuzzy Information in Fuzzy FOIL

The fundamental data types of Fuzzy FOIL include the usual Prolog terms, namely integers, real numbers, constants, and lists. In addition, fuzzy sets represented as linguistic terms can be used where there is vagueness. For example, linguistic term "high salary" can be considered as a fuzzy set which can be characterized by a membership function with

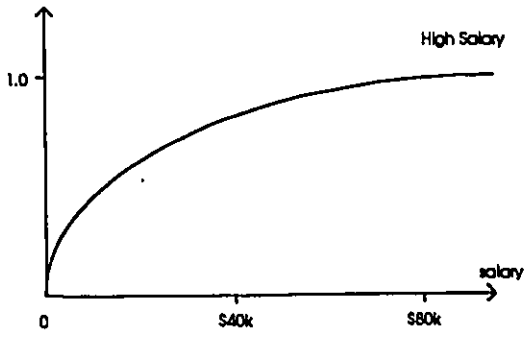


figure 4.3.1

possibility distribution as figure 4.3.1.

The fact that 'Tom has high salary' can then be represented as a Prolog fact clause:

salary(tom, #high_salary).

Here, the prefix symbol '#' is used to indicate that the term following is a fuzzy linguistic term and that there is a corresponding membership function in the knowledge base. Fuzzy linguistics can be manipulated by standard set operations. In addition, they allow uncertain inferences to be made when fuzzy linguistic terms are partially matched by fuzzy unification. For example, if it is known that a person named Tom has high salary, the result of query to find people with a salary of "about $45000" would be Tom and a truth degree of support. This degree is calculated by matching two fuzzy terms "high_salary" and "about $45000".

3.4.2 Mixed Data Types in Fuzzy FOIL

In FOIL, the data type of a term which is either numerical data (continuous-value) or symbolic data, must be the same in all the examples in the training set. Two data items of different types are not allowed to be the values of the same attribute. This constraint on data type limits the application of FOIL to 'ideal' data domains. In real life domains, however, there exists a great deal of vague data, which can never be represented in exact and well-defined form. In many cases, the collection of available information is naturally found to be described in different data type. Mixed data types encountered in learning systems arise frequently. By incorporating fuzzy theory, these mixed data type are allowed in the training set in Fuzzy FOIL. For example, in the following data set of information about employee's salary, some employee's salary is known exactly and can be represented in quantitatively as "$60000". In some other cases, instead of specifying a person's salary as quantitative value, one can be specified as "about $58000" or " a good salary".

salary(tom, #'high_salary').

salary(smith, 60,000).

salary(bob, #'about58000').

salary(bush, #'good_salary').

salary(tang, 70,000).

So in the above facts, numerical values 60,000 and 70,000 can be mixed with symbolic values 'high_salary' and 'good_salary' in the same attribute 'salary' in a ground fact. All these symbolic values are informative and imprecise statements that would be found to be more convenient in people's communication in our every day life. It is essential for machine learning system to learn in imprecise and vague data domains.

3.4.3 The Possibility of Positiveness in Example.

FOIL assumes everything in given examples must be exactly known. An example must be either absolutely positive or definitely negative. However, in real life, these assumptions would lead to the following three problems,

First, it often happens that problem solving and decision making by humans are performed in environments where information concerning the problem is partial or approximate. For example, if you are not quite sure about an answer to the question 'Are you still going to swim if it's raining tomorrow?', you may naturally respond to it with an uncertain answer 'possibly' or 'maybe not'. This raises the question: what if this sort of uncertainty answer is to be used as a part of a training example in ILP, and how do current ILP systems handle this? The answer is that using such uncertain data is beyond the scope of the current ILP systems.

Second, by hard classification of examples into the positive or negative ones, misclassification is always possible. Soft classification of examples with degree of positiveness or negativeness would reduce the chance of misclassification [Kacprzyk and Iwanski 1993].

Third, if everything is exactly known, why should we still learn something? If learning is only possible under the condition that everything must be exactly told, and there is no capability to learn from the environment with uncertainly known facts, the usefulness of learning system is degraded.

To overcome these limitations, in Fuzzy FOIL a fact that stands for a positive (or negative) example is allowed to be partially true (or false) to some degree of positiveness (or negativeness), respectively. This not only provides an approach to represent the imprecision in the classification of examples but also reduces the chance of misclassification.

In fuzzy Logic, possibility can be treated as a linguistic variable whose truth-values form a term set as below:

f(Possibility) = { definite , impossible , almost_definite , almost_impossible,

very_possible , fairly_possible ,...}

The degree of positiveness(or negativeness) can be expressed as either linguistic value described as above or numerical values in the interval [0, 1]. And we have the following form of example:

e(T1, T2, ... ,Tn) : f(Possibility)

Suppose we have a set of examples about the shoe size people wear and people's height. We try to let learning system find out the definition of *wear_large_shoes* from these examples by the relation between people's height and their shoe size. The example is formed as :

wear_shoes(Name, Height, Shoes'size): degree_of_possibility.

e.g.,

wear_shoes(tom, 195, 12): very_possible.

wear_shoes(smith, #short, #big_size): 0.0.

wear_shoes(bob, #very_tall, #very_big_size): definite.

wear_shoes(mary, 179, 13): 0.6.

3.5 Background Knowledge

Background knowledge in Fuzzy FOIL consists of three possible models:

- A set of extensional background predicate definitions defined as function-free fuzzy horn ground facts.
- A set of intensional definite-clause concept definitions which entails the positive examples and negative examples
- Fuzzy linguistic term model which is a library of fuzzy membership functions. A fuzzy membership function is defined as Prolog clause.

Fuzzy linguistic term is a value of fuzzy linguistic variable which has been introduced in chapter 2.

For example, Zadeh describes TALL as a linguistic variable, which represents our cognitive category of "tallness". To each person in the universe of discourse, we have to assign a degree of membership in the fuzzy subset TALL. The easiest way to do this is to define a membership function based on the person's height (this is just one example of a possible membership function).

$$\mu_{tall}(x) = \begin{cases} 0, & x =< 155 \\ \dfrac{1}{1+e^{(58.95-0.35x)}} & 155 < x < 180 \\ 1, & x >= 180 \end{cases}$$

It can be defined as a predicate definition in Fuzzy FOIL:

```
fuzzy(#tall,X,Degree):-
        X < 155, Degree = 0.0.
fuzzy(#tall,X,Degree):-
        X > 155,X < 180,
        exp((58.9 - 0.35*X), Degree).
fuzzy(#tall,X,Degree):-
        X >= 180, Degree = 1.0.
```

Given this definition, here are some example values:

Person	Height	degree of tallness
Billy	154	0.00
Yoke	168	0.33
Drew	174	0.79
Erik	179	0.96
Mark	182	1.00

Expressions like "A is X" can be interpreted with a degree of truth,

e.g., the possibility degree of "Drew is TALL" is 0.79

Membership functions used in most applications almost never have the shape as simple as tall(x). At the minimum, they tend to be triangles pointing up, and they can be much more complex than that. We will discuss it in detail at chapter 6 on how to construct membership functions from training data.

4. Theoretical Foundations of Fuzzy Specialization Technique

Specialization techniques search the hypothesis space top-down, from the most general to specific concept descriptions. Since the top-down search can easily be guided by heuristics, specialization techniques are better suited for inductive learning in the presence of imperfect data. The theoretical basis of the specialization technique used in FOIL is a θ-subsumption based refinement operator. Under θ-subsumption, a refinement operator typically computes only the set of minimal (most general) specialization of a clause. It applies two syntactic operations on a hypothesis clause:

- apply a substitution to the clause to substitute a variable with a term.
- add a literal to the body of a clause.

In Fuzzy FOIL, when fuzziness of given examples and background knowledge is taken into account, traditional definition of θ-subsumption in FOIL can not be directly applied. FOIL is meant to be used in "ideal" domains in which given examples and bias knowledge are assumed to be exact, precise, and well-defined. Therefore, just as θ-subsumption is essential to traditional refinement operator, a new definition of θ-subsumption, which will structure Fuzzy-Prolog based hypothesis search space, is necessary for fuzzy logic based specialization techniques. In the following section, we will give a new definition of θ-subsumption and extension so that Fuzzy FOIL has a theoretical foundation to support its implementation of extending FOIL's ability to learn under real-world domains where data and background knowledge are often imperfect and uncertain.

4.1 Structuring the Fuzzy Hypothesis Space - Fuzzy θ-Subsumption

A learning system can be described in terms of the structure of its hypothesis search space, its search strategy, and its search heuristics. Therefore, inductive learning can be viewed as a search problem. This section introduces the fuzzy θ-subsumption lattice, which determines the structure of the search space of hypothesis fuzzy-horn clauses and is a basis for introducing a partial ordering into hypothesis clauses. To define fuzzy θ-subsumption, a notion of fuzzy substitution needs to be defined first. We extend traditional substitution and θ-subsumption notions as follows.

Definition 4.1: fuzzy-substitution

A substitution $\theta = \{X1/t_1,...,Xn/t_n\}$ or $\{\#FV_1/t_1, ..., \#FVn/t_n\}$ is a function mapping variables ***Xi*** or fuzzy linguistic term ***#FVi*** to terms t_i. Term t_i can be either a pure Prolog term or a fuzzy linguistic term which was described in previous chapter. The application $W\theta$ of a substitution θ to a wff ***W*** is obtained by replacing all occurrences of each variable ***Xi*** or ***#FVi*** *in* ***W*** by the same term t_i if and only if:

f(**fuzzy_match**(*#FVi*, t_i)) ≥ f(e(..t_i..)).

where f(**fuzzy_match**(*#FVi*, t_i)) means the degree of matching fuzzy linguistic values with term t_i , and f(e(..t_i..)) represents the degree of positiveness of the given example tuple which contains term t_i.

Definition 4.2: fuzzy θ-subsumption

Let **c** and **c′** be two fuzzy horn clause. Clause **c** θ-subsumes **c′** if there exists a substitution θ, such that $c\theta \subseteq c'$ and $f(c\theta) \leq f(c')$, where $f(c\theta)$ and $f(c')$ are truth

degree of two clauses. Two clauses **c** and **d** are θ-subsumption equivalent if **c** θ-subsumption **d** and **d** θ-subsumption **c**, and $f(\mathbf{c}\theta) = f(\mathbf{d}\theta)$. A clause is reduced if it is not θ-subsumption equivalent to any proper subset of itself.

In the above definition, an additional constraint $f(\mathbf{c}\theta) \leq f(\mathbf{c}')$ is introduced. The following discussion will prove why this new constraint must be applied in the definition of fuzzy θ-subsumption.

In the fuzzy context, as the definition of fuzzy sub set in chapter 2, fuzzy set A is a subset of B if and only if for every element x in A and B based on the same universal U, the degree of membership $\mu_A(x) \leq \mu_B(x)$, i.e.

$$A \subseteq B \text{ iff } \forall x \in U,\ \mu_A(x) \leq \mu_B(x)$$

Now, **c** and **c′** can be considered as two fuzzy sets, by applying the above fuzzy set definition, if there exits a substitution θ, such that $\mathbf{c}\theta \subseteq \mathbf{c}'$ only and only if for all elements in these both fuzzy sets, the degree of fuzzy membership of **c**θ must less or equal to the degree of fuzzy membership of **c′**. That is, $f(\mathbf{c}\theta) \leq f(\mathbf{c}')$. Therefore this additional constraint to the definition of fuzzy θ-subsumption is necessary.

Example 4.1

To explain the above fuzzy θ-subsumption definition, let us discuss the following examples.

Suppose fuzzy FOIL is used to learn concept *strong man* and let **c** be the clause:

c: strong_man(V_name) :- candidate(V_name, V_sex, V_age), V_age = #young.

Under empty substitution $\theta = \phi$, we declare that clause **c** fuzzy θ-subsumes clause **c'**:

c': strong_man(V_name) :- candidate(V_name, V_sex, V_age), V_age = #about_26.

Since fuzzy set {V_age = #about_26} is a subset of fuzzy set { V_age = #young }, as shown in Figure 4.1 i,e.

$$\{V\_age = \#about\_26\} \subseteq \{ V\_age = \#young \},$$

By applying negative operator, we have fuzzy set relation:

$$\{ \neg V\_age = \#young\} \subseteq \{\neg V\_age = \#about\_26\}$$

Therefore, the memberships of these two fuzzy set have the following relation:

$$f(\{ \neg V\_age = \#young\}) \leq f(\{\neg V\_age = \#about\_26\})$$

This leads to that fuzzy set:

{ strong_man(V_name) $\vee$ $\neg$candidate(V_name, V_sex, V_age) $\vee$ $\neg$ V_age = #young} is a fuzzy subset of fuzzy set:

{ strong_man(V_name) $\vee$ $\neg$candidate(V_name, V_sex, V_age) $\vee$ $\neg$ V_age = #about_26}

i.e., we have proven that $\mathbf{c}\theta \subseteq \mathbf{c}$.

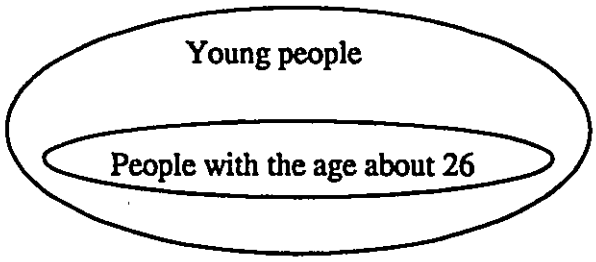


Figure 4.1

Now, by the definition 2 of fuzzy θ-subsumption, we also need to prove that

$$f(\mathbf{c}\theta) \leq f(\mathbf{c}').$$

As discussed in chapter 2, the union $A \cup B$ of two fuzzy sets A and B is a fuzzy set in U with the membership defined for all $u \in U$ by

$$\mu_{A \cup B}(u) = max\{\mu_A(u), \mu_B(u)\}$$

By applying the above fuzzy maxim operator:

$f(\mathbf{c}\theta) =$

$f(\{\text{strong\_man(V\_name)} \vee \neg\text{candidate(V\_name, V\_sex, V\_age)} \vee \neg\ \text{V\_age} = \#\text{young}\})$

$=$

$\max(f(\{\text{strong\_man(V\_name)}\}), f(\{\neg\text{candidate(V\_name, V\_sex, V\_age)}\}), f(\{\neg\text{V\_age} = \#\text{young}\}))$

which is less or equal to:

$f(\mathbf{c}') =$

$f(\{\text{strong\_man(V\_name)} \vee \neg\text{candidate(V\_name,V\_sex, V\_age)} \vee \neg\ \text{V\_age} = \#\text{about\_26}\})$

$=$

$\max(f(\{\text{strong\_man(V\_name)}\}), f(\{\neg\text{candidate(V\_name,V\_sex,V\_age)}\}), f(\{\neg\text{V\_age} = \#\text{about\_26}\}))$

Since we have proven that:

$f(\{\neg\text{V\_age} = \#\text{young}\})) \leq f(\{\neg\text{V\_age} = \#\text{about\_26}\})$.

therefore, $f(\mathbf{c}\theta) \leq f(\mathbf{c}')$ is satisfied in this example.

Both $\mathbf{c}\theta \subseteq \mathbf{c}'$ and $f(\mathbf{c}\theta) \leq f(\mathbf{c}')$ are both satisfied. That is, **c** fuzzy θ-subsumes clause **c**.

Based on the definition 1 of fuzzy substitution, the next example will show further the difference between traditional θ-subsumption and fuzzyθ-subsumption.

Suppose, there exists a positive example with degree of positiveness 0.95,

⊕ strong_man(tom,male, 28): 0.95

To have a fuzzy linguistic term substituted, fuzzy substitution definition requires that the truth degree of matching fuzzy linguistic term with replaced term should be greater or at least equal to the degree of positiveness of the examples that include replaced term. So, the first step is to invoke fuzzy match function to get value of degree:

fuzzy_match(#young, 28) = 1.0

since f(strong_man(tom,male, 28)) = 0.95 has been given , so we have

f(**fuzzy_match**(#young, 28)) > f(strong_man(tom,male, 28))

Then fuzzy linguistic term #young in the clause **c** is substituted by numeric term 28, i.e., we have substitution,

θ = {V_name/tom, V_sex/male, #young/28}.

Under this substitution, clause **c:**

strong_man(V_name) :- candidate(V_name, V_sex, V_age), V_age = #young.

θ-subsumes clause **c'**:

strong_man(tom) :- candidate(tom, male, V_age), V_age= #about_26

Because, f(**fuzzy_match**(#young, 28)) =1.0 > f(**fuzzy_match**(#about_26, 28)) = 0.85

and { strong_man(tom) , ¬candidate(tom, male, #young), ¬ #young = 28}

is a subset of

{strong_man(tom) , ¬candidate(tom, male, V_age), ¬V_age= #about_26}

4.2 Properties of fuzzy θ-Subsumption

fuzzy θ-subsumption provides the basis for top-down searching of refinement graphs and ensures that the refinement process computes only the set of the minimal specialization (most general) and only the highest truth degree of clauses. Since

incorporating fuzzy truth degree with each hypothesis clause. According to fuzzy θ-subsumption principle, a modified notion of generality and specialization is described as:

Definition 4.3: Fuzzy Generalization and Specialization

Clause **c** is called at least as general as clause **c′** (**c** ≤ **c′**) if **c** θ-subsumes **c′** and $f(\mathbf{c}) \leq f(\mathbf{c}')$. Clause **c** is more general than **c′** (**c** < **c′**) if **c** ≤ **c′** holds and **c′** ≤ **c** does not. In this way, we also have that **c′** is a specialization of **c**.

Based on the above definition, we have an important property of fuzzy θ-subsumption: **The more general the clause, the less fuzzy truth degree; the more specialized the clause, the higher fuzzy truth degree.**

This property of fuzzy θ-subsumption directs Fuzzy FOIL search form most general to specific hypothesis and finally finds a maximal general (minimal specialization) clause with the highest belief degree.

4.3 Fuzzy Coverage : Completeness and Consistency

The fundamental theoretical definitions of θ-subsumption and generality are purely syntactic notions, since they do not take into account any background knowledge. To make them operational for inductive learning, the notion of fuzzy coverage is introduced. Before defining fuzzy coverage, we need to extend the definition of traditional inductive logic learning in the fuzzy context.

Fuzzy Logic based Inductive Learning

Given a set of training examples $E = \{E^+, E^-\}$ (E^+ represents positive examples and E^- denotes a set of negative examples), background knowledge B, and fuzzy membership function library ***fDB***. E, B, and H are all expressed in fuzzy horn clause. E stands for a

set of positive examples or negative examples which are allowed to be expressed as partly true to some degree of positiveness or negativeness. The aim is then to find a hypothesis H, such that the following conditions hold:

$B \cap E^- \not\models_{fDB} \phi$

$B \cap H \not\models_{fDB} \phi$

$B \cap H \models_{fDB} E^+$

The procedure of coverage checking plays an important role in the above formulation of fuzzy inductive concept learning.

Definition 4.4: Fuzzy Coverage

Given background knowledge B, membership function library fDB, hypothesis H, and example set E, hypothesis H fuzzy covering example $e \in E$ with respect to background knowledge B is constrained by the following conditions,

if $B \cup H \vdash e$ and $f(\text{cover}(B, H, e)) \geq f_e(e)$ for some of $f \in fDB$

where predicate cover/2 is used for coverage checking and, $f(\text{cover}(B, H, e)) \in [0, 1]$, $f_e(e) \in [0, 1]$. $f_e(e)$ denotes the degree of possibility of positiveness of given example e.

The traditional definitions of completeness and consistency requirements are also extended as:

Definition 4.5: Completeness

A hypothesis H is complete with respect to background knowledge B and positive examples E^+ if all the positive examples are covered (maximal generalization), i.e.,

$\text{cover}(B, H, e) = true, e \in E^+$

and $f(\text{cover}(B, H, e)) >= f_d(e)$

Definition 4.6: Consistency

A hypothesis H is consistent with respect to background knowledge B and examples E if no negative examples are covered (minimal specialization). There are two cases when no negative examples are covered:

- negative example e can not be deduced from $B \cup H$. *i.e* $B \cup H \not\models e$
- negative example is deduced from $B \cup H \models e$, but $f(\text{cover}(B \cup H, e)) < 1 - f\_n(e)$

where $f\_n(e)$ represents the possibility degree of negativeness of example e. The second case means that, for each negative examples $e \in E$- with possibility of negativeness: $f\_n(e) \in [0, 1]$, no negative example e is covered by the hypothesis clause H if the degree of hypothesis clause covering a negative example is less than the degree of being positiveness of a negative example.

The above definition can be also mathematically expressed as:

$\text{cover}(B, H, e) = \phi$

if $B \cup H \not\models e$ or $B \cup H \models e$ and $f(\text{cover}(B \cup H, e)) < 1 - f\_n(e)$

Since hypothesis language is Fuzzy-Prolog, Fuzzy FOIL uses Prolog resolution and unification techniques to check whether an example can be deduced from $B \cup H$. Therefore the above notion of coverage is an intensional coverage. The background knowledge and given example can contain both ground facts and non-ground fuzzy horn clauses. On the other hand, FOIL uses the extensional notion of coverage which requires

extensional background knowledge in the form of ground facts only. In the case that background knowledge consists of intensional predicates, it has to be transformed into ground facts. For a Prolog based system, comparing the two approaches of coverage checking, extensional coverage has some serious disadvantages. Since clauses are learned extensionally, but then the whole program is interpreted by Prolog intensionally, in general, extensional approach is not adequate, as it requires listing of all the necessary examples.

5. Fuzzy Specialization Algorithm

This section presents a fuzzy empirical Inductive Learning Algorithm which is based on the top-down search of refinement graphs introduced in FOIL. The Top-down search of refinement graphs is a basic specialization ILP technique which employs a specialization operator to repeatedly refine hypothesis clause from the most general clauses until there is no negative example covered by hypothesis clause. In Fuzzy FOIL, the specialization refinement operator is redefined as follows.

5.1 Fuzzy Specialization Refinement Operator

Given a fuzzy horn clause based language bias fL, a refinement operator ρ refines a clause c to a set of clauses $\rho(c)$ by applying fuzzy θ-subsumption. The finite set of $\rho(c)$ is called specialization of c:

$$\rho(c) = \{ c' \mid c' \in fL, c < c' \,\&\, f(c) \le f(c') \}$$

where $f(c)$ and $f(c')$ reflect the degree of specification or the truth degree of hypothesis clause c and c'. The higher the truth degree of a clause, the more specialized (less general) clause is.

It employs two basic operations on a hypothesis clause:

- Add a literal to the body of the clause.
- Apply a fuzzy substitution to the clause.

Having defined the fuzzy specialization refinement operator, the process of learning can be performed by iterative searching of refinement graph. Before introducing a fuzzy

top-down refinement graph algorithm, the following basic notation and terminology need to be explained first:

- A hypothesis H is a set of fuzzy horn clauses, i.e. $H = \{C \mid C \in fL\}$. The clause and hypothesis currently built by the system are called the current clause, denoted as ***Ccur*** and the current hypothesis, denoted as ***Hcur*** respectively. The current clause ***Ccur*** is formed as $G \leftarrow [f]\text{-}Q$. The head G is an predicate term $goal(X1,...,Xn)$, where ***goal*** is the target relation symbol, f is the implication strength of $G \leftarrow Q$, and body Q is a conjunction of literals $L1, ..., Lm$.
- The set $E = \{ e{:}f(e) \mid E^{+} \cup E\text{-}\}$, is the given training set. It consists of positive examples E^{+} and negative examples E-. Each unit example tuple represents one of the following three cases:

  1) predicate_name(T1, T2, ... Tn).

  A positive example with definite truth degree of positiveness.

  2) not(predicate_name(T1, T2, ..., Tn)).

  A negative example with definite truth degree of negativeness.

  3) predicate_name(T1, T2, ... , Tn):f(e)

  An uncertain type of positive example with degree of positiveness f(e).

  4) not(predicate _name(T1,T2, ...,Tn):fn(e))

  An uncertain type of negative example with degree of negativeness fn(e).

  Term Ti contained in the example can be one of three data types:

  - continuous numeric value,
  - symbolic constant,

- a fuzzy concept or fuzzy set represented as fuzzy linguistic symbolic value.

When constructing the current clause ***Ccur***, the current training set is denoted as ***Ecur***. A fuzzy top-down refinement algorithm now can be outlined in the following section.

5.2 Fuzzy Top-Down Refinement Graph Algorithm

The fuzzy algorithm consists of two main sub algorithms: covering algorithm and refinement operation algorithm. Based on AQ [Michalski] and CN2[Clark and Niblett], the covering algorithm performs hypothesis construction and the refinement operation algorithm performs clause specialization/construction.

The covering algorithm performs a repeat coverage checking loop. A search for a hypothesis clause starts with an empty body, the most general clause and with the truth degree or possibility value of 0.0. It constructs a hypothesis in three main steps :

- construct a clause,
- add the clause to the current hypothesis, and
- remove the positive examples E^+ covered by the current hypothesis clause ***Hcur*** from the current positive examples set ***Ecur***.

This loop is repeated until all the positive examples are covered.

The refinement operation algorithm is performed by repeatedly applying fuzzy refinement operator ρ. It constructs current hypothesis clause: $Ccur = G \leftarrow (f) - Q$ by adding a most promising literal to the body of ***Ccur*** or by substituting variable and fuzzy linguistic term in the current clause with new term. Similar to the hill-climbing search, the fuzzy refinement graph algorithm keeps the "best" clause and replaces it with its "best" refinement at each specialization step, until the stopping criterion is reached. The

search heuristic method used in Fuzzy FOIL is a modified information gain method of FOIL. We will discuss it in the next section. The complete fuzzy top-down specialization algorithm is illustrated in the following:

Algorithm 5.1 Fuzzy Specialization Algorithm

Initialize local training set $Ecur = E$.

Initialize $H = \phi$.

Initialize the first hypothesis clause to $Ccur = goal(V1,..,Vn) \leftarrow \{0.0\}$-

while $Ecur^{+} \neq \phi$ // covering

while $Ecur^{-} \neq \phi$ { if there exists negative $e\Theta$ that $f(cover(Ci, e\Theta)) >= f(e\Theta)$ }
// refining
Find the best literal $Lbest$; add it to the body $G \leftarrow \{f\}$-Q :

$$Ccur = G \leftarrow \{f\}\text{-}Q, Lbest.$$

Form a new local training set $Enew$ as set of extensions of the tuples in $Ecur$ that satisfy $Lbest$:

$$f(cover(Lbest, e\oplus)) \geq f(e\oplus)$$

endwhile

Calculate the implication strength f of $Ccur = G \leftarrow \{f\}$- $Q1,Q2, \ldots Qn$.

Add new constructed clause to hypothesis set:

$$H' = H \cup Ccur$$

Remove all tuples that satisfy H':

$$Enew = Ecur - cover(B, H', Ecur^{+})$$

endwhile

Output Hypothesis H.

5.3 Heuristic Method in the Fuzzy Refinement Graph

The search strategy used in Fuzzy FOIL is hill-climbing. The heuristic for directing the searching is a modified statistic information-based method called possibility associated information gain. The basic idea is to select candidate literal with the highest possibility information gain.

The possibility associated information gain used in Fuzzy FOIL is described as follows: Suppose that T is current set of example, T^+ is a set of the current positive examples, and T- is a set of current negative examples. Let $T^{\oplus}$ denote the sum of the possibility degree of all positive tuples covered by current clause in T. Let $T^{\ominus}$ denote the sum of the possibility degree of all negative examples covered by current clause in T, i.e.:

$$T^{\oplus} = \sum f(\text{cover}(H, e)) ; e \in T^+$$

$$T^{\ominus} = \sum f(\text{cover}(H, e)) ; e \in T-$$

Then the information used to discriminate the negative and positive examples for hypothesis clauses H is,

$$I(T) = -\log_2[T^{\oplus}/(T^{\oplus} + T^{\ominus})]$$

When the next literal *Lbest* is selected, a new training set T′ based on new hypothesis H′ is also created. The information measure then can be calculated by:

$$I(T') = -\log_2[T'^{\oplus}/(T'^{\oplus} + T'^{\ominus})]$$

where $T'^{\oplus}$ and $T'^{\ominus}$ represent the sum of possibility degree of each positive example and negative example covered by a new clause in T′ respectively. Suppose T^{++} is the set of positive examples in T that are represented by one or more examples in T′ , then the information gained by selecting literal Lbest amounts to:

$$\mathbf{Gain}(\mathit{Lbest}) = |T^{++}| \times (I(T) - I(T'))$$

Therefore, by taking the degree of example covering into account, Fuzzy FOIL not only selects the literal with the highest gain, but also selects the literal with the highest possibility of positive examples covering.

6. Fuzzy Membership Function Construction

One of the most powerful insights of L.A.Zadeh's fuzzy theory is the observation that fuzzy linguistic terms, such as 'short', 'almost', 'few', 'many', can be represented by functions whose values are numerical degree of membership in the membership domains. Such a representation is fundamental to the modeling of fuzzy learning system and is an essential key element of any natural language in human communication.

6.1 View on Representation of Fuzzy Membership

In the current literature, there are three different, but related views on the representation of membership functions.

Vertical representation: a fuzzy set A on a referential set X can be viewed as a mapping μ_A (known as the membership function) from X to [0, 1], i.e., $\mu_A(X) \rightarrow [0, 1]$, where each element $x \in X$ belongs to A to a degree. $\mu_A(x) = 1$ means full membership and $\mu_A(x) = 0$ means non-membership in the fuzzy set A. This is called the "*vertical*" representation of a fuzzy set A [Dubois and Prade 1989]. It views X as the horizontal axis of a planar representation and puts the unit interval on a vertical axis (Figure 6.1).

Horizontal representation: a fuzzy set A can be represented in terms of its level sets $\{A_\alpha \mid \alpha \in (0, 1]\}$, where $A_\alpha = \{x \mid \mu_A(x) \geq \alpha\}$, such that

$\mu_A(x) = \sup\{\alpha \in (0, 1] \mid x \in A_\alpha\}$.

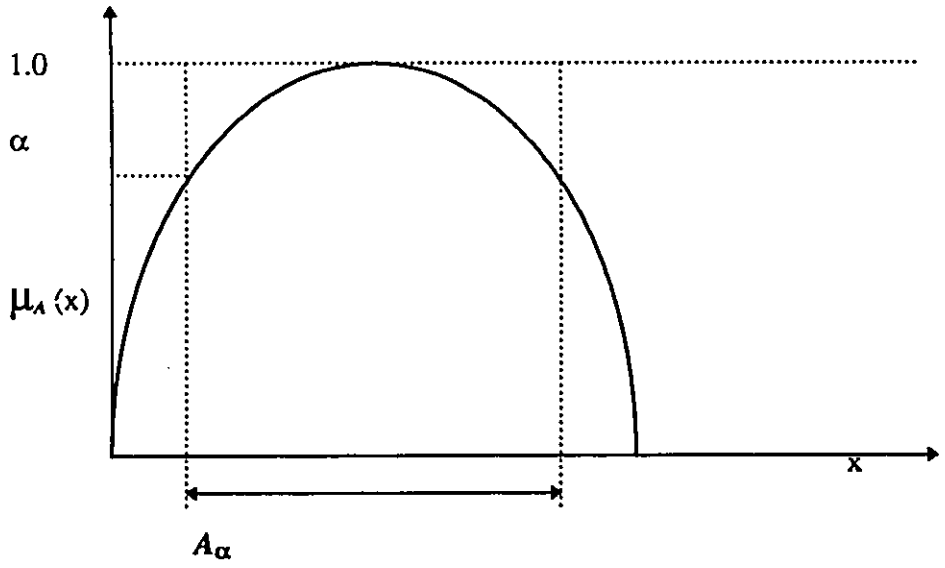


Figure 6.1

Given a family of sets $\{A_\alpha \mid \in (0, 1]\}$ with the condition that $\alpha \geq \alpha'$ implies $A_\alpha \supseteq A_{\alpha'}$, such a family is known as the set of level-sets of a fuzzy set A. This is called the "*horizontal*" representation of a fuzzy set as a nested family of level-sets by [Dubois and Prade] (See figure 6.1).

Random set based representation: suppose we have a finite discrete fuzzy set A, with its level sets $\{ A_\alpha \mid \alpha \in (0, 1]\}$. Let $M(A) = \mu_A(X)\text{-}\{0\} = \{ \alpha 1 > \alpha 2 > \ldots \alpha n\}$ be the set of n positive possible membership values. Also, define $mi = \alpha i - \alpha i+1$, and let $\alpha n+1 = 0$ by convention. Then, we can define:

$$\mu_A(x) = \sum_i mi \text{ such that } x = A\alpha i.$$

$\{(A, m_i) \mid i = 1, \ldots, n\}$ is usually called a 'random set' where 'random' is a misleading name. It should more appropriately be called a convex combination of (ordinary) sets. It is now known that this representation helps us to understand the connection between fuzzy

set theory and the theory of 'evidential reasoning'. In fact, it has been shown that each fuzzy set A defines an equivalence class of 'random sets' whose contour function is the membership function μ_A.

In the following sections, based on the view of *"vertical"* representation of a fuzzy set, the various types of modeling fuzzy membership function will be discussed and a linear regression based approach to model membership functions in our system will be presented.

6.2 Types of Modeling of Membership Functions

Approximating a Fuzzy Concept

When we turn to build fuzzy models, fuzzy systems are tolerant of approximations not only in their problem spaces but also in the representation of fuzzy concepts (sets). This means that they will perform well even when the fuzzy set is an approximation of the concept [Cox, Earl 1994]. The fuzzy membership functions change the fuzzy set curve surface. Generally, but not always, the surface is a continuous line from the left to the right edge of the set. The shape of a fuzzy set represents the semantics properties of the underlying fuzzy concept, Therefore, the closer the set surface maps to the behavior of a conceptual phenomenon, the better our model will reflect the real world.

Modeling a fuzzy membership function can be done in a number of ways. Here, we look at the "standard" fuzzy shape used most often in fuzzy models - lines, S-curves, Z-curves , and π- shapes.

Lines shape function

The linear proportionality surface is a straight line. It is a good choice and perhaps the simplest fuzzy set when approximating an unknown or poorly understood concept. There are two kind of a linear fuzzy set: the increasing set and the decreased set. The increasing set starts at a domain value that has zero degree of membership in the set and moves to the right with values that have increasing degree of membership. As an example, see "Hot" fuzzy set membership function (Figure 6.2a).

The decreasing fuzzy set is the opposite of the increasing set as example "Cold" fuzzy set in (Figure 6.2b).

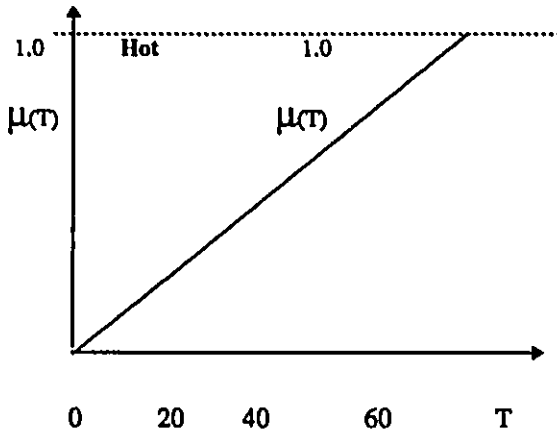


Figure 6.2a

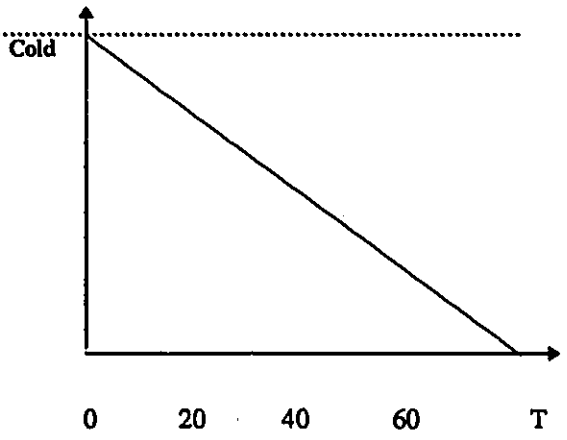


Figure 6.2b

S-Curve , Z-Curve, and π-Curve membership function

The 'Tall' and 'Slow' fuzzy concept membership functions are typical examples of S-curve and Z-curve respectively. They correspond to the increasing and decreasing nonlinear surfaces. A growing S-curve set moves from no membership at its zero-degree

membership at its extreme left-hand side to full-degree at its extreme right-hand side. Figure 6.2c illustrates the shape of a growth S-curve.

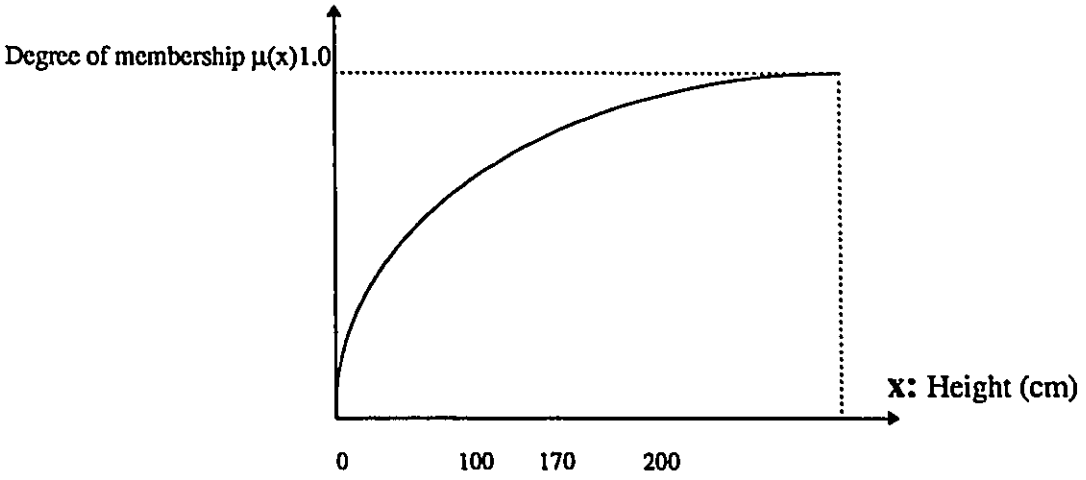


Figure 6.2c A growth of S-curved fuzzy set : Tall

The declining Z-curve set moves from complete membership at its extreme left-hand side to zero membership at its extreme right-hand side. Figure 6.2d illustrates a decline Z-curve.

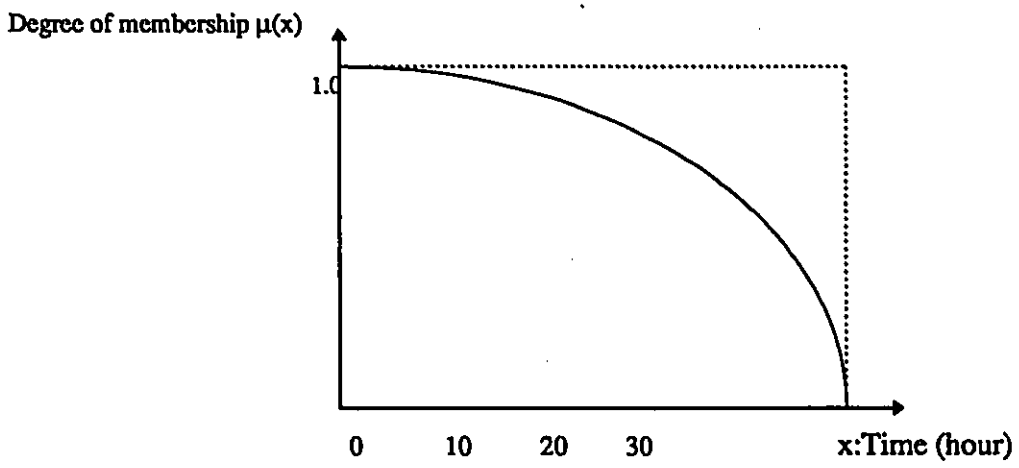


Figure 6.2d A decline Z-curved fuzzy set : Slow

π-Curve shape fuzzy set is an important class of fuzzy contours representing approximates of a central value and are graphically visualized as a class of π-Curve shapes. The concept of *Middle_Aged* is a good example of π-Curve fuzzy set. In the crisp set point of view, concepts *Young* and *Middle Aged* are two crisp sets. We are required to find some exact transition points where an individual's age moves him or her from the set *Young* into the set *Middle Aged* and then from this into the set *Old*. In fuzzy set we consider that someone 40 years old represents the expected value of middle-age fuzzy set since every one would believe that this individual is middle aged. A π-Curve shape as Figure 6.2d is used to map individual age into the degree of membership of *Middle Aged* concept. This π-Curve function spreads out the degree of Middle Ageness on both sides of the 40-years-old domain point. This corresponds to the general concept - some 30 years old has a significant but not intense degree of middle Ageness. One the other side of π-Curve hill, someone 50 years old is still considered to some degree of middle aged.

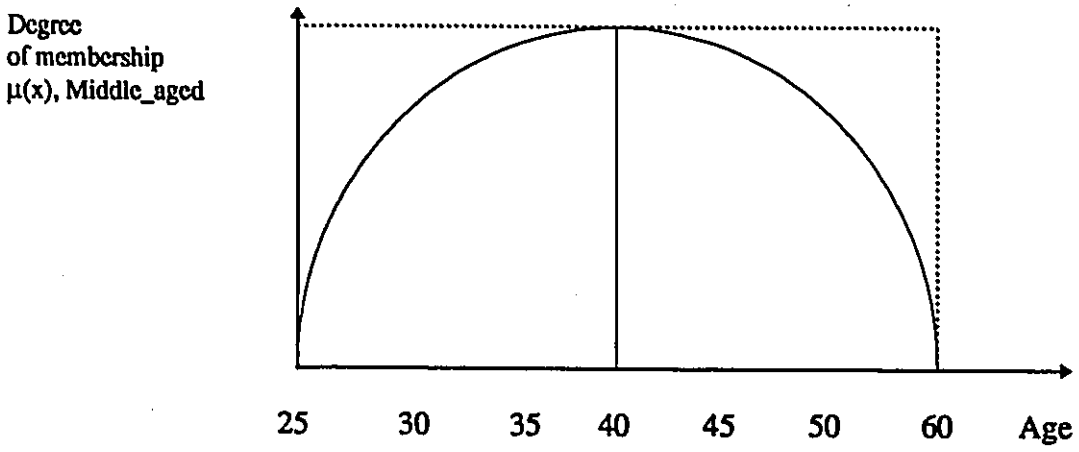


Figure 6.2d π-Curve membership function

6.3 A Linear Regression Based Membership Construction Approach

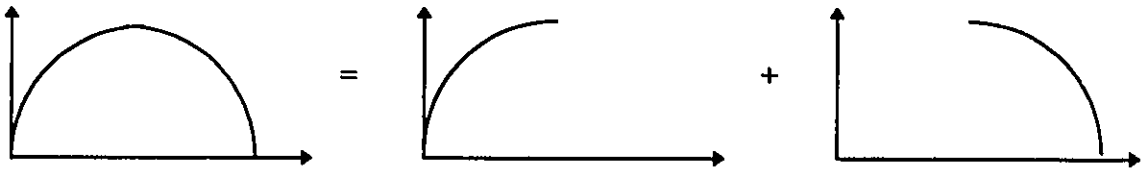
Once the membership properties have been defined, the next issue is how to model a function. That is, how to estimate the membership function curves. Traditionally, a membership function can be derived from statistical data or be approximately determined based on human expert's opinion or human's common perception. In the current literature, the "three points method" is the most popularly accepted approach for modeling a membership function. It characterizes S-curve using three parameters: its zero membership value (α), its full degree complete membership value (γ), and crossover point (β) which is the point at which the domain value is 0.5 degree of membership. The membership function is defined as [Cox, 94],

$$\mu(x: \alpha, \beta, \gamma) = \begin{bmatrix} 0 & x \le \alpha \\ 2((x-\alpha)/(\gamma-\alpha))2 & \alpha \le x \le \beta \\ 1-2((x-\gamma)/(\gamma-\alpha))2 & \beta \le x \le \gamma \\ 1 & x \ge \gamma \end{bmatrix}$$

For π-curve function, a membership function can be assumed as a triangular shape function. The slopes of the triangular membership function are selected so that adjacent membership functions cross at the membership value 0.5. Then the centers parameters can be calculated by using Kohonen's feature-maps algorithm [Kohonen, 1988] (the details of the algorithm can be found in [Yufei, 1994]).

In our system, we use an approach which is based on linear regression technique to generate all types of curve functions.

Firstly, we consider that 'π-curve hill' can be represented as one S-curve shape on its left-side hill and one Z-curve shape on its right -side hill as shown in Figure 6.3.



π-curve S-curve Z-curve

Figure 6.3

Since Z-curve function can be defined as 1 - S-curve function, we only need to focus on modeling a well defined S-curve function. If S-curve function is well modeled, Z-curve and π-curve membership function will be consequently well modeled based on the definition of S-curve membership function.

To construct a S-curve function, we suppose further that S-curve function is a continuous differentiable shape and derivative $\mu'_s(x)$ is jointly proportional to $\mu_s(x)$ and $(1-\mu_s(x))$, where $\mu'_s(x)$ represents the degree of membership in " X is S-curve" and $(1-\mu_s(x))$ represents X is not S-curve" [Kochen, 1974]. This means that the marginal increase in the judged strength of belief that $x \in$ a fuzzy set S is proportional to both the strength of the belief that $x \in S$ and the strength of his belief that $x \notin S$, or , assumed to be 1-$\mu_s(x)$, i.e.,

$$\mu'_s(x) = d(\mu_s(x))/dx = c\mu_s(x)(1-\mu_s(x))$$

with the solution:

$$\mu_s(x) = (1 + e^{a-bx})^{-1} \quad \Rightarrow \ln[\mu_s(x)^{-1} - 1] = a - bx$$

the formula $\ln[\mu_s(x)^{-1} - 1] = a - bx$ can be considered as,

$$y(x) = \ln[\mu_s(x)^{-1} - 1], \Rightarrow y(x) = a - bx$$

As this is a linear formula, two parameters a and b can be estimated by the well-known linear regression technique.

7. Experimental Results

To show that Fuzzy FOIL is a powerful and general mechanism for learning under uncertainty environment, this section presents the results obtained by using Fuzzy FOIL on a variety of learning tasks.

7.1 Learning *Strong_Man* Concept Definition

The first example is to show the workings of Fuzzy FOIL on learning a new fuzzy logic concept. The concept *Strong_Man* is a fuzzy concept since it can not be exactly defined in terms of physical features of individuals. To learn a new concept strongMan(Name), this example uses one relation candidate as domain background knowledge,

candidate(Name,Sex,Age,Height).

This relation provides information about an individual's basic physical features: Name, Sex, Age and Height.

Fuzzy FOIL constructs hypothesis clauses based on the following given training examples:

```
/* Positive examples */

strongMan(smith):-{definite}.
strongMan(john):-{1.0}.
strongMan(martin):-{0.99}.
strongMan(tang):-{0.99}.
strongMan(hao):-{0.75}.
strongMan(jie):-{0.15}.
strongMan(bob):-{0.3}.
strongMan(mary):-{impossible}.
strongMan(anna):-{0.15}.
strongMan(tom):-{0.3}.
strongMan(susan):-{0.48}.

/* Background Knowledge */
candidate(smith,male,20,198).
```

```
candidate(john,male,#young,200).
candidate(martin,male,#about25,190).
candidate(tang,male,26,#about190).
candidate(hao,male,40,195).
candidate(jie,male,45,159).
candidate(bob,male,26,#short).
candidate(mary,female,#about65,#tall).
candidate(anna,female,45,197).
candidate(tom,male,65,197).
candidate(susan,female,25,#tall).

foil_theory_constant([[candidate,2,male,female],[candidate,3,#young,#middle_aged],
                      [candidate,4,#tall]]).
```

Two constant symbolic terms: male and female and three fuzzy linguistic terms: #young , #middle_aged, and #tall, are used as theory constants which are introduced and are explained in Quinlan's FOIL system and the corresponding papers [Quinlan 1993, 1990]. The fuzzy linguistic terms are supported by corresponding fuzzy membership functions which are considered as a part of background knowledge in Fuzzy FOIL.

In this learning task, negative examples are generated automatically according to 'Closed-World' assumption. The result is quite satisfactory. Fuzzy FOIL always tries to find new concept with highest possibility(degree of truth). The first found clause definition with highest possibility is constructed as:

```
strongMan(A) :-[0.99501]-
      candidate(A,B,C,D),
      B= male,
      D= #tall,
      C= #young.
```

This is a fuzzy horn clause with degree of truth 0.9951. It defines concept *Strong_Man* as a person who is a man, is tall and young. The truth degree 0.9951 reflects this definition's belief degree or possibility level.

The second definition of *Strong_Man* with lower truth degree is defined as the following relation,

```
strongMan(A) :-[very_possible/0.75]-
      candidate(A,B,C,D),
      C= #middle_aged,
      D= #tall,
      B= male.
```

It states that a person can be very possibly considered as a strong man if he is in the middle age and certainly this person must be a male person.

A comparison between above learned fuzzy classes and the clauses induced by FOIL is given below:

Fuzzy FOIL:

```
strongMan(A) :-[0.99501]-
      candidate(A,B,C,D),
      B= male,
      D= #tall,
      C= #young.
strongMan(A) :- [0.75]-
      candidate(A,B,C,D),
      C= #middle_aged,
      B= male,
      D= #tall.
```

FOIL:

```
strongMan(A) :-
      candidate(A,B,C,D),
      C<=41,
      B=male,
      D>161.
```

We can see that result induced from Fuzzy FOIL is more specific than FOIL's. The key difference is that Fuzzy FOIL not only can discover more specific new knowledge which involves fuzzy linguistic terms but also tells you what degree of truth that discovered knowledge has and what degree of belief you can associate with the new knowledge.

7.2 Learning *Close_friend* Relationships

The second example is to learn the definition of *close_friend* relationship. This example shows that Fuzzy FOIL has advantage of dealing with continuous attribute value over FOIL. This task involves four relationships:

close_friend(Name1,Name2).
% Name1 and Name2 are names of two closed friend.

relationship(Name1,Name2,Relation).
% Two persons Name1 and Name2 have a Relation.

candidate(Name,Sex,Age).
% Basic background information about personal name, his/her sex and age.

hobby(Name, Hobby).
% A person's hobby information.

The target relation is close_friend(Name1,Name2), which states that person Name1 and person Name2 are two close friends, in terms of the background knowledge relations candidate, hobby, and relationship. Fuzzy FOIL learns the *close_friend* from the following background knowledge and training examples.

```
/* Positive examples */
close_friends(anna, mary):-{0.8}.
close_friends(smith,susan):-{0.785}.
close_friends(thomas,henry):-{0.789}.

/* Negative examples */
close_friends(wang,tom):-{0.1}.
not(close_friends(smith,mary)).
not(close_friends(smith,henry)).
not(close_friends(anna,susan)).
not(close_friends(anna,henry)).
not(close_friends(thomas,susan)).
not(close_friends(thomas,mary)).
not(close_friends(thomas,fred)).

/* Background Knowledge */
foil_theory_constant([[relationship,3,friend],[candidates,3,#young]]).
```

```
relationship(smith,susan,friend).
relationship(smith,mary,couple).
relationship(smith,henry,relative).
relationship(anna,susan,relative).
relationship(anna,mary,friend).
relationship(anna,henry,friend).
relationship(thomas,henry,friend).
relationship(thomas,mary,friend).
relationship(thomas,susan,relative).
relationship(thomas,fred,relative).
relationship(wang,tom,friend).

candidates(smith,male,23).
candidates(anna,female,22).
candidates(mary,female,25).
candidates(susan,female,27).
candidates(thomas,male,20).
candidates(henry,male,24).
candidates(fred,male,25).
candidates(tom,male,75).
candidates(wang,male,65).

hobby(smith,painting).
hobby(anna,cooking).
hobby(mary,cooking).
hobby(susan,painting).
hobby(thomas,fishing).
hobby(tom,fishing).
hobby(wang,fishing).
hobby(henry,fishing).
hobby(fred,fishing).
```

Fuzzy FOIL starts with specializing current clause as:

```
close_friends(A,B) :-
        candidates(A,C,D),
        candidates(B,E,F),
        relationship(A,B,G),
        hobby(A,H),
        hobby(B,I).
```

This clause covers three positive examples, but it is too general in that it covers eight negative examples as well. Next, Fuzzy FOIL tries to find a more specialized clause, by substituting variable *H* with *I* in clause hobby(A,H). hobby(A,H) is a more specialized clause which reduces the number of covered negative examples from eight to two:

```
close_friends(A,B) :-
        candidates(A,C,D),
        candidates(B,E,F),
        relationship(A,B,G),
        hobby(A,H),
        hobby(B,I),
        hobby(A,I).

Covered 2 negatives:
[(close_friends(wang,tom):0.1),(close_friends(thomas,fred):0.0)]
```

Finally, Fuzzy FOIL adds new theory constants to further constrain clause. Three new theory constants G=friend, F=D= #young are added. The final induced clause then is constructed as:

```
close_friends(A,B) :-[0.79133]-
        candidates(A,C,D),
        candidates(B,E,F),
        relationship(A,B,G),
        hobby(B,I),
        hobby(A,I),
        G=friend,
        F= #young,
        D= #young.
```

This logic definition of close_friends relation can be explained as having the truth degree of 0.79133, two candidates are *close-friend* only if they at least are in the friend relation (exclude any relative relation), and have one same hobby, and are both young.

This learning *close_friend* experiment is also conducted with FOIL on the same training data set as used by Fuzzy FOIL. But FOIL failed to induce a complete definition. FOIL got stuck at an uncompleted clause:

```
close_friends(A,B) :-
    hobby(A,C),
    candidates(A,D,E),
    candidates(B,F,G),
    relationship(A,B,H),
    hobby(B,C),
    H=friend,
```

The key point here is that FOIL was unable to find a comparison of the continuous-valued variables of *E* and *G* with threshold to represent an interval value of *age=young*. Since the values of variables representing candidate's age are not strictly in order, FOIL has difficulty to find a threshold from those continuous-values. In contrast, Fuzzy FOIL elegantly transforms continuous-valued features into a set of fuzzy categories which define the fuzzy semantics associated to real-world term: *#young*. In this way, Fuzzy FOIL learns relation from uncertainty data and imprecise data by using inexact matching and approximate inductive reasoning. However, for FOIL, you have to provide it with exact and complete data. No uncertainty and inaccuracy are allowed in FOIL. So, for this experiment learning task, FOIL finally concludes with:

No literals

Clause too inaccurate (3/4)

This example shows that Fuzzy FOIL significantly extends FOIL's ability to deal with uncertainty data and the ability of handling continuous-value.

7.3 Learning Fuzzy Shower Control Rules

Let us illustrate in more detail the performance of Fuzzy FOIL by a more complicated example - a task that learns shower control rules from a fuzzy shower simulator. This task will demonstrate that Fuzzy FOIL can be applied in practical applications. The fuzzy shower simulator was used as an example of Fuzzy CLIPS developed in Institute for Information Technology of Canadian National Research Council.

The fuzzy shower simulator simulates the flow and temperature of water leaving a shower head as a function of time and a built-in fuzzy controller that keeps the flow and temperature within some required ranges. As Figure 7.3, the flow out of the shower head (Fx) at any time t is exactly equal to the flow of cold water (Fc) at time t plus the flow of hot water(Fh) at time t. Any change to a valve position (Vc, Vh) is immediately reflected in the flow from the shower head. There are 25 fuzzy control rules used in the simulator for adjusting the hot and cold water taps to keep flow and temperature in the target range. The fuzzy rule is represented as Fuzzy CLIPS's rule form as the following:

```
(defrule OK_strong
(outTemp OK)
(outFlow strong)
=>
(assert (change_vh NS))
(assert (change_vc NS))
)
```

This rule means if out temperature is OK (a fuzzy term) and out flow is too strong (a fuzzy term) then change hot valve in small negative value and change cold valve in small negative value.

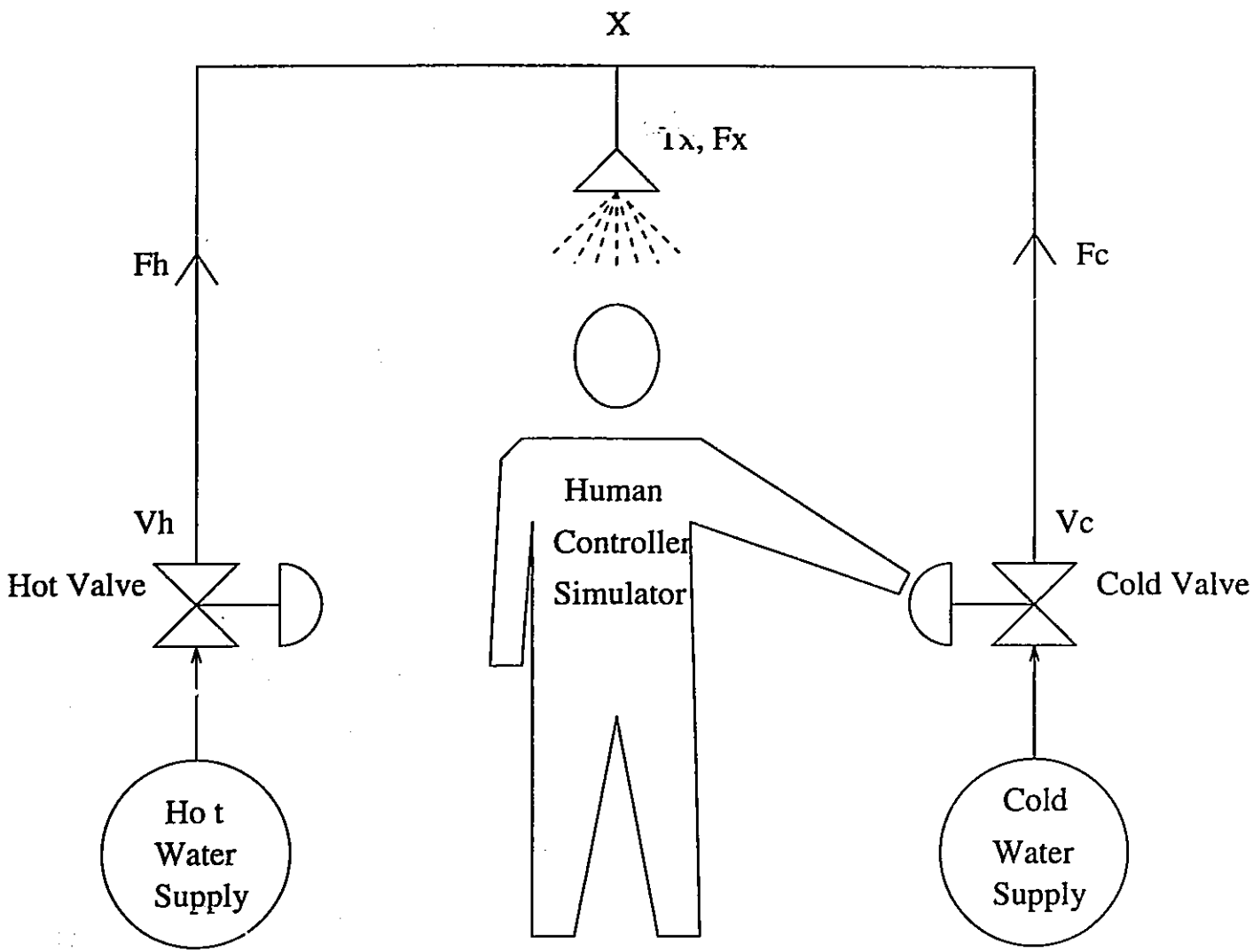


Figure 7.3 Fuzzy Shower Controller

The fuzzy shower controller simulator was used to demonstrate Fuzzy CLIPS's application and fuzzy approximate reasoning ability. Here, we use it to collect the output of the simulator as training and testing data for learning shower controller rules inside the simulator. In this sense, the simulator is considered as a black box, Fuzzy FOIL tries to guess the fuzzy control rules built in the box based on collected input and output data from the simulator.

The task of learning Fuzzy Shower Control Rule can be formulated as: given collected output of temperature, flow of water, and the change values of both hot and cold valves from simulator, to find shower control rules that control the shower simulator.

Training data were collected by playing the simulator on the Web page of Institute for Information Technology of Canadian National Research Council (at URL: http: // ai.iit.nrc.ca/fuzzy/shower/title.html).

We produce random initial hot water value and cold water value as initial input for the shower simulator. Then Fuzzy CLIPS rules inside the simulator take over control to adjust the shower's hot tap and cold tap to get the shower running in target zone for both temperature and flow. The target temperature is between 34 and 38 degrees Celsius and the target flow rate is between 11 and 13 liters/sec. We record adjusted hot value, cold value, corresponding flow, and temperature output values as positive training examples.

The following examples records show a complete process of Fuzzy CLIPS rules in the simulator tuning hot and cold valves to get the shower running in target zone.

Flow	Temper	ColdValve Change	HotValve Change
32.4310,	35.0320,	-0.1256,	-0.1232
24.3240,	35.0630,	-0.0900,	-0.0890

18.4610,	35.0790,	-0.0370,	-0.0380
15.9970,	35.0800,	-0.0290,	-0.0280
14.1610,	35.0820,	-0.0220,	-0.0230
12.6660,	35.0840,	0.0000,	0.0000

where the last record shows that the shower output target range has been reached. That is flow value 12.6660 is between 11 and 13 liters/sec, and temperature value 35.0840 is between 34 and 38 degrees Celsius, and ColdValve, HotValve has zero change.

As Fuzzy FOIL allows for the use of non-ground background knowledge and training data, we directly use the Fuzzy Prolog's definition of negative example predicates:

```
not(shower_control_rule(#'fOK', #cold,#'NM',not(#'PM'))).
not(shower_control_rule(#'fOK', #cold,not(#'NM'),#'PM')).
not(shower_control_rule(#'fOK', not(#cold),#'NM',#'PM')).
not(shower_control_rule(not(#'fOK'), #cold,#'NM',#'PM')).
```

where #cold, #fOK, #'NM' (means negative medium change), and #'PM' are fuzzy theory constants. Given the 5564 positive examples, 100 negative examples as described above, Fuzzy FOIL generated 43 different control rules, ten of learned rules with higher truth degree are shown below (the complete rules are listed in Appendix A).

Found definition:

```
shower_control_rule(Flow,Temp,ColdV,HotV) :-
      HotV= #somewhat_Zero,
      ColdV= #somewhat_Zero,
      Temp= #tOK.
With Truth Degree of : 0.544879

shower_control_rule(Flow,Temp,ColdV,HotV) :-
      HotV= #somewhat_Zero,
      ColdV= #somewhat_Zero,
      Flow= #fOK.
With Truth Degree of : 0.406634
```

```
shower_control_rule(Flow,Temp,ColdV,HotV) :-
      Flow= #strong,
      HotV= #somewhat_Zero,
      ColdV= #somewhat_NS,
      Temp= #cool.
With Truth Degree of : 0.776099

shower_control_rule(Flow,Temp,ColdV,HotV) :-
      Flow= #strong,
      ColdV= #somewhat_Zero,
      HotV= #somewhat_NB,
      Temp= #hot.
With Truth Degree of : 0.823739


shower_control_rule(Flow,Temp,ColdV,HotV) :-
      Flow= #strong,
      HotV= #somewhat_NS,
      ColdV= #somewhat_Zero,
      Temp= #tOK.
With Truth Degree of : 0.715906

shower_control_rule(Flow,Temp,ColdV,HotV) :-
      Flow= #strong,
      HotV= #somewhat_NS,
      ColdV= #somewhat_Zero,
      Temp= #warm.
With Truth Degree of : 0.759082

shower_control_rule(Flow,Temp,ColdV,HotV) :-
      Flow= #strong,
      HotV= #somewhat_Zero,
      ColdV= #somewhat_NB,
      Temp= #cold.
With Truth Degree of : 0.789931

shower_control_rule(Flow,Temp,ColdV,HotV) :-
      Flow= #strong,
      ColdV= #somewhat_NM,
      HotV= #somewhat_Zero,
      Temp= #cool.
With Truth Degree of : 0.973207

shower_control_rule(Flow,Temp,ColdV,HotV) :-
      HotV= #somewhat_Zero,
```

```
    ColdV= #somewhat_PS,
    Temp= #tOK,
    Flow= #little.
With Truth Degree of : 1.0

shower_control_rule(Flow,Temp,ColdV,HotV) :-
    Flow= #strong,
    HotV= #somewhat_NM,
    ColdV= #'Zero',
    Temp= #hot.
With Truth Degree of : 0.836265
```

The first two rules state that if the temperature and flow value are *OK*, there is *"Zero"* change (no change) in both of the hot water valve and the cold water valve. These two rules also mean that the once shower output target range has been reached, no change on both hot and cold valves are needed. The last rule states that if flow output is *strong* and temperature is *hot*, then the action should be to change the hot valve in somewhat negative medium value (HotV = #somewhat_NM) and no change in the cold valve . Two factors affect the above results induced by Fuzzy FOIL:

- The way that fuzzy shower simulator produces simulated temperature and flow of water is to union output (a fuzzy set) from each Fuzzy CLIPS rule, and this fuzzy union set is defuzzified as a crisp value. Those crisp value data are used as training examples for training learner (Fuzzy FOIL), so it is impossible for the learner to construct a rule which is exactly like the rule in the shower simulator. This factor also reduces the truth degree of learned rules induced by Fuzzy FOIL.
- The example data we have collected from the shower simulator are not sufficient to cover whole hypothesis search space. We are currently trying to collect more data for Fuzzy FOIL to learn more rules.

The same training examples were used to train FOIL. FOIL found the following definitions of the shower control rules:

shower_control(A,B,C,D) :- A>14.611, D>-0.149, D<=-0.0007, C<=-0.0007.
shower_control(A,B,C,D) :- C<=0, D>-0.0002, A>12, A<=13.001.
shower_control(A,B,C,D) :- B<=34.03, D>0.0001, A>10.49, D<=0.058, A<=13.447, C<=-0.001.
shower_control(A,B,C,D) :- A<=14.611, B>34.03, B<=37.968, A>13.343, C<=-0.007, D<=-0.003.
shower_control(A,B,C,D) :- D<=0.002, C<=0.0005, A>28.718, B<=45.911, B>14.498.
shower_control(A,B,C,D) :- A<=14.611, D<=0.002, C>0.003, A>10.601, C<=0.066, B>37.988.
shower_control(A,B,C,D) :- C<=0.0035, C>-0.0005, D<=0, D>-0.001, A>10, A<=11.996.
shower_control(A,B,C,D) :- C<=0.0035, D<=0.003, A>13.447, D>0.0001, A<=13.498.
shower_control(A,B,C,D) :- A<=14.765, A>13.498, C<=-0.001, D<=-0.001.
shower_control(A,B,C,D) :- A<=14.862, A>12.998, B>34.251, B<=37.356, C<=-0.004, D<=-0.003.
shower_control(A,B,C,D) :- A<=14.862, C<=0.0005, D<=0.0003, C>0.0001.

For comparing Fuzzy FOIL and FOIL, the ideal approach would be to have their learned rules evaluated by applying these rules to control Fuzzy shower simulators. We need to set up three simulators which are controlled by Fuzzy CLIPS rules, rules learned from Fuzzy FOIL, and rules learned from FOIL, respectively. The same initial settings for the shower will be selected for the three shower simulators. The three simulators will be activated to adjust the shower to make it under control and to get shower running in target zone for both temperature and flow. We can then evaluate Fuzzy FOIL and FOIL against Fuzzy CLIPS. The evaluation on Fuzzy FOIL and FOIL will be measured by the percentage of their rules successfully getting shower running in the target zone, and by the number of steps used to make shower reach the target zone.

However, rules induced by Fuzzy FOIL and FOIL can not be directly used in the shower simulator, since rules used in the shower simulator should be a set of Fuzzy CLIPS programs while rules induced by Fuzzy FOIL and FOIL are fuzzy Prolog and pure prolog

clauses. Therefore, a shower simulator that can use Fuzzy prolog and prolog clause need to be implemented first for the evaluation. At the current time the Fuzzy prolog based shower simulator is still under development. The experimental evaluation results will be soon presented in the paper followed to the thesis.

To have a general evaluation of the results for Fuzzy FOIL and FOIL on learning shower control rules, a simplified evaluation method is used. Under this evaluation method, the rules from both Fuzzy FOIL and FOIL are matched with 50 test cases collected by running the fuzzy shower simulator. We define a set of complete records of data, which represent every step that fuzzy CLIPS rules successfully control the shower simulator to reach the target zone as a test case. Like training examples shown in page 62, each record in a test case corresponds to a step that fuzzy CLIPS rules control the hot and the cold valve of the shower. It consists of the change values of both cold and hot valve, temperature of water, and flow of water at the time that the Fuzzy CLIPS rules are activated. The evaluation is measured by the number of test cases "exactly matched" with learned rules, the number of test cases "properly matched" with learned rules, and the number of test cases "no matched" with learned rules.

"exactly matched" requires that for each record in a test case, there is a matched corresponding learned rule. It means that if applying learned rules in the test case, the shower simulator will be controlled to reach target zone as the way it is controlled by the Fuzzy CLIPS rules at exactly same number of steps.

"properly matched" means that each record in a test case might not have a matched tested rule, but at least the first record (initial status of the shower) and the last record (the target zone) in the test case should be matched with the tested rules.

"no matched" means any cases excluding "exactly matched" and "properly matched".

The result of simplified evaluation is listed below:

	"exactly matched"	"properly matched"	"no matched"
Fuzzy FOIL	20	19	11
FOIL	0	0	50

One of reasons that FOIL fails to have better results in this learning task is its limited ability to learn relation from large continuous-values examples. Another reason is that, FOIL uses extensional coverage method, it needs examples and background to be represented in ground fact. Fuzzy FOIL uses intensional coverage. Background knowledge and given example in Fuzzy FOIL contain both ground facts and non-ground fuzzy horn clauses. Since negative examples in this learning task are represented as non-ground clauses for Fuzzy FOIL, we need to transfer those non-ground fuzzy horn negative examples to ground facts completely to guarantee that we have the exact same examples for both two systems, so that the comparison of the two system can be fairly conducted. However, this transformation can not guarantee that deduced ground facts is exactly same with original non-ground clause in terms of the meaning and coverage. Therefore, the negative examples for FOIL might differ from that used by Fuzzy FOIL. This factor could affect the result of FOIL.

8. Conclusions and Future Work

In this thesis research, we do not restrict ourselves to horn clause based pure logic. By incorporating fuzzy logic into Inductive Learning Programming, we present a new method to solve problems arising in inductive logic learning tasks in the domains affected by uncertainty and vagueness. Our system is the first one that investigates the approach of combing inductive logic programming with fuzzy logic programming for improving the applicability of ILP in the real world domains. The applicability of the method has been demonstrated on three learning tasks. The results show that our system has definite advantages over the traditional ILP system FOIL in dealing with uncertainty imperfect data and numerical continuous data that are often encountered in real applications.

8.1 Summary of work

This thesis presents a system called Fuzzy FOIL that extends FOIL's ability of handling numerical data and uncertain data. Our work includes the implementation of an executable experimental system using Quintus-Prolog 3.1.1 and some important fundamental theoretical basis work which have been established to support the system implementation. The following list summarizes the contributions of Fuzzy FOIL and major research work of this thesis.

1. By discussing the difference between possibility and probability, we point out that statistical noise is not the only source of imperfect data. Other types of uncertainty, vagueness and inexactness, are often mistreated as probability uncertainty. Therefore, we suggest fuzzy theory as an alternative to probability theory for dealing with possibility uncertain imperfect data.

2. Pure Prolog used as hypothesis language in FOIL has a serious shortcoming: its inability to cope with the issue of uncertainty and imprecision. Since pure Prolog has this limitation, current Prolog based ILP systems usually do need an extra process called "pre-processing of data and background knowledge" to change fuzzy, imprecise data and knowledge to an accurate and precise form so that they are ideal enough for system to deal with. This extra process not only keeps ILP away from practicable application, but also, loses important information represented by fuzzy data. A more powerful representation language Fuzzy Prolog is introduced as the hypothesis language used in our system to overcome the limits of pure Prolog.
3. Current ILP system assumes everything in examples and background knowledge must be either absolutely positive or definitely negative and must be exactly known. However, real domain data are often naturally imprecise and uncertain. In our system, any example or background knowledge can be inexact. Uncertainty can have degree between [0, 1]. So, our system can handle any uncertain and vague data properly.
4. By applying inexact match or fuzzy unification, another improvement on representing real domain data has been implemented. That is, mixed data type is allowed in the same attribute in an example fact so that we do not need specify clearly the type of data for each attribute in the example and the background knowledge.
5. By introducing fuzzy substitution and fuzzy θ-subsumption concepts, our system has a solid theoretical foundation to support its implementation and future extension.

6. By redefining the notion of completeness and consistency, the hypothesis clause learned by our system is guaranteed to be complete and consistent with respect to given examples and background knowledge.
7. The three learning tasks presented in this thesis have shown that the introduction of *fuzzy theory constant* has improve FOIL's ability in handling continuous-valued data. The key advantage of Fuzzy FOIL is that it proves Fuzzy Theory to be suited to transform continuous-valued features into a set of fuzzy categorical attributes and to define the fuzzy semantics associated to real-world terms and predicates. In this way, it extends FOIL 's ability to deal with uncertainty data and the ability of handling continuous-valued attributes.
8. The example of learning 'fuzzy shower controller rules' demonstrates that Fuzzy FOIL can be applied in real-life applications, especially in the domains characterized by imprecise and uncertain information, such as control applications. By combining Fuzzy Logic, the Inductive Learning techniques can be easily applied to control tasks, which extends the scope of application domains. Process control seems to be a good area of fuzzy logic application.
9. The results produced by Fuzzy FOIL are easily understood by people, and have more comprehensible concepts.
10. A possibility associated information gain method is introduced to guide heuristic search during fuzzy refinement process. It guides learner to select a highest possibility hypothesis clause from candidates under uncertainty environment.

8.2 Limitations

1. Current Fuzzy FOIL inherits FOIL's ability to learn recursive clauses, but there is an unsolved issue of learning fuzzy horn recursive clauses.
2. Performance of Fuzzy FOIL is not so efficient as FOIL since dynamic inexact matching is used to support fuzzy substitution.

8.3 Future Work

There are may potential improvements and further researches on our system. In this thesis work, we have focused on establishing a fundamental fuzzy inductive logic programming framework. Based on this introductory framework, many interesting research can be carried out:

1. A good fuzzy inference based stopping criteria is necessary in our system. This can improve the efficiency of system performance.
2. An algorithm for pruning fuzzy horn clause also is a very interested topic.
3. A larger research area is the fuzzy generalization technique. It includes number of research topics, such as fuzzy reverse resolution, fuzzy semantic θ-subsumption and so on.

Reference

Botta, M., Giordana, A. and Saitta, A. *Learning Fuzzy Concept Definitions*. 10th International Conference on Machine Learning. Amherst, MA, p18-22, 1993.

Brunk and Pazzani (1991), *An investigation of noise-tolerant relational concept learning algorithms*. In Proc. Eighth International Workshop on Machine Learning, pages 389 - 393, Morgan Kaufmann, San Mateo, CA. 1991.

Cesnik, B. and Bratko, I. (1991) . *On estimating probabilities in tree pruning*. Proc. Fifth European Working Session on Learning, pages 151-163. Springer, Berlin. 1991.

Charles, Elkan (1993). *The paradoxical Success of Fuzzy Logic*. Proceeding of the Eleventh national Conference on AI (AAAI'93). pp.698-703, 1993.

Chen, Guiming (1993). *A Prototype of fuzzy-Prolog Interpreter*. Tech. Report. Dept. of Computer Sci. Memorial University of Newfoundland, 1993.

Clark, P. and Niblett, T. (1989) *The CN2 induction algorithm*. Machine Learning, 3(4):pages 261-283, 1989.

Clark, P. and Boswell,R. (1991). *Rule induction with CN2: Some recent improvements*. In Proc. Fifth European Working Session on Learning, pages 151-163. Springer, Berlin, 1991.

Clark, Peter & Matwin, Stan (1993). *Using Qualitative Models to Guide Inductive Learning*. 10th International Conference on Machine Learning. Amherst, MA, p49-56, 1993.

Cox, Earl (1994). *The Fuzzy System Handbook: a practitioner's guide to building and maintaining fuzzy systems*. Boston, Mass., AP professional. 1994.

Dubois, D. and Prade, H. (1989) *Fuzzy Sets, probability and measurement*. European J. Oper. Res. 40 1989 pages 1350-154, 1989.

Dzeroski (1991). *Handling noise in inductive logic programming*. Master's thesis, Faculty of Electrical Engineering & Computer Science, Univ. of Ljubljana, Slovenia, 1991.

Dzeroski, S. & Lavrac, N (1991). *Learning Relation From noise examples: an empirical comparison of LINUS and FOIL.* In Proceeding of the Eighth International Workshop on Machine Leering, pages 399-402, 1991.

Kacprzyk, J. and Iwanski (1993). *Inductive Learning From Incomplete and Imprecise Examples.* In MLW-93, 1993.

Klir, George J. (1989). *Is There More To Uncertainty Than Some Probability Theorists Might Have Us Belive?* Int. J. General Systems, Vol. 15. pages 347-378, 1989.

Kochen, Manfred. (1974). *On the Precision of Adjectives which Denote Fuzzy Sets.* J. Cybernetcs, 1974,4,1 pages 49-59, 1974.

Kohonen, T. (1988). *Self-Organization and Associative Memory* . Springer, Berlin, 1988.

Kosko, Bart (1990). *Fuzziness .vs. Probability.* Int. J. General Systems, Vol. 17. pages 211-240, 1990.

Lavrac, N. and Dzeroski, S. (1992). *Inductive learning of relations from noisy examples.* Inductive Logic Programming, Edited by Stephen Muggleton, Academic Press, UK. pages 495-514, 1992.

Lavrac, N. and Dzeroski, S. (1994). *Inductive Logic Programming - Techniques and Applications.* Ellis Horwood Limited, UK, 1994.

Li, Deyi , Liu, Dongbo and Chen, Guiming.(1989). *A New Fuzzy Inference Language f-PROLOG.* Computer Engineering. Jan. 1989, pages 23-27

Li, Deyi and Liu, Dongbo. (1990). *A Fuzzy Prolog Database System.* Research Studies Press LTD, UK, 1990.

Matwin, Stan & Rouget, Thierry (1996). *Explainable Induction with an Imperfect Qualitative Model.* to be submitted, available at URL: http://www.csi.uottawa/~stan/my_research.html.

Matwin, Stan & Stan Szpakowicz (1992). *Machine Learning Techniques in Knowledge Acquisition from Text.* THINK 1:2, 37-50.

Muggleton,S. and Buntine, W. (1988). *Machine invention of first-order predicates by inverting resolution.* In Proc. Fifth International Conference on Machine Learning, pages 339-352. Morgan Kaufmann, San Mateo, CA, 1988.

Muggleton,S. and Feng, C. (1990). *Efficient Induction of Logic Programs.* In Proc. First Conference on Algorithmic Learning Theory, pages 368-381, Ohmsha, Tokyo, 1990.

Muggleton, S., Srinivasa, A., and Bain, M. (1992). *Compression, significance and accuracy.* In Proc. Ninth International Conference on Machine Learning, pages 338-347. Morgan Kanufmann, San mateo, CA, 1992.

Pazzani et al. (1991). *A knowledge-intensive Approach to Learning Relational Concept.* In Proc. Eighth International Workshop on Machine Learning, pages 432-436.

Quinlan,I. (1993). *C4.5:Programs for Machine Learning.* Morgan Kaufmann, San Mateo, CA.

Quinlan, J.R. (1990). *Learning Logical Definitions from Relations.* Machine Learning, 5 (pages 239 - 266), 1990.

Quinlan, J.R. and Cameron-Jones (1993). *FOIL: A Midterm Report,* in Brazdil P.B.(ed), Machine Learning: ECML-93, Springer, Berlin, pages. 3- 20, 1993.

Quinlan, J.R.(1986) *Induction of decision trees.* Machine Learning, 1(1): 1986 pages 81-106

Srinivasa, A., Muggleton, S., and M.Bain (1992). *Distinguishing conceptions from noise in non-monatomic learning.* In Proc. Second National Workshop on Inductive Logic Programming. Tokyo, Japan ICOT TM-1182, 1992.

Yager, R.R. (1982). *Some Procedures for Selecting Fuzzy Set Theoretic Operators. Int. J. of General Systems,* 8:115-124, 1982.

Yuan,Yufei and Shaw, Michael J.(1995) *Induction of fuzzy decision trees.* Fuzzy Sets and Systems 69, 1995.

Zadeh, L.A. (1975). *The Concept of a Linguistic Variable and its Application to Approximate Reasoning*. Int. Sci. 8, 1975.

Zadeh, L.A. (1965). *Fuzzy Sets*. Inform. and Control. 8, 1965, pages 338-353.

Zadeh, L.A. (1978) *PRUF - A Meaning Representation Language for Natural Languages*. J. of Man-machine Study. 10, 1978b, pages 395-460

Zadeh, L.A. (1984). *Fuzzy Probabilities*. Inf. Processing and Management 19, 1984, pages 148-153, 1984.

Zadeh, L.A. (1988). *Fuzzy Logic*. Computer 1, pages 83-93, 1988.

Zadeh, L.A.(1978) *Fuzzy Sets as a Basis for a Theory of Possibility*. J. of Fuzzy Set Systems 1, 1978a, pages 3-28, 1978.

Appendix A: The Result of Fuzzy FOIL on Learning Fuzzy Shower Control Rules

Found definition:

```
shower_control_rule(Flow,Temp,ColdV,HotV) :-
      HotV= #somewhat_Zero,
      ColdV= #somewhat_Zero,
      Temp= #tOK.
With Truth Degree of : 0.544879

shower_control_rule(Flow,Temp,ColdV,HotV) :-
      HotV= #somewhat_Zero,
      ColdV= #somewhat_Zero,
      Flow= #fOK.
With Truth Degree of : 0.406634


shower_control_rule(Flow,Temp,ColdV,HotV) :-
      Flow= #strong,
      ColdV= #somewhat_NS,
      HotV= #somewhat_NS,
      Temp= #tOK.
With Truth Degree of : 0.504628

shower_control_rule(Flow,Temp,ColdV,HotV) :-
      Flow= #strong,
      ColdV= #somewhat_NS,
      HotV= #somewhat_NS,
      Temp= #hot.
With Truth Degree of : 0.0807728

shower_control_rule(Flow,Temp,ColdV,HotV) :-
      Flow= #strong,
      ColdV= #somewhat_NS,
      HotV= #somewhat_NS.
With Truth Degree of : 0.243136


shower_control_rule(Flow,Temp,ColdV,HotV) :-
      Flow= #strong,
      HotV= #somewhat_Zero,
      ColdV= #somewhat_NS,
      Temp= #cool.
With Truth Degree of : 0.776099

shower_control_rule(Flow,Temp,ColdV,HotV) :-
      Flow= #strong,
      ColdV= #somewhat_Zero,
      HotV= #somewhat_NB,
      Temp= #hot.
With Truth Degree of : 0.823739

shower_control_rule(Flow,Temp,ColdV,HotV) :-
```

```
      HotV= #somewhat_Zero,
      ColdV= #somewhat_NS,
      Flow= #high,
      Temp= #cool.
With Truth Degree of : 0.433985

shower_control_rule(Flow,Temp,ColdV,HotV) :-
      Flow= #strong,
      HotV= #somewhat_NS,
      ColdV= #somewhat_Zero,
      Temp= #tOK.
With Truth Degree of : 0.715906

shower_control_rule(Flow,Temp,ColdV,HotV) :-
      Flow= #strong,
      HotV= #somewhat_NS,
      ColdV= #somewhat_Zero,
      Temp= #warm.
With Truth Degree of : 0.759082

shower_control_rule(Flow,Temp,ColdV,HotV) :-
      HotV= #somewhat_Zero,
      ColdV= #somewhat_NS,
      Temp= #tOK,
      Flow= #strong.
With Truth Degree of : 0.632674

shower_control_rule(Flow,Temp,ColdV,HotV) :-
      HotV= #somewhat_Zero,
      ColdV= #somewhat_NS,
      Flow= #high.
With Truth Degree of : 0.301487

shower_control_rule(Flow,Temp,ColdV,HotV) :-
      Flow= #strong,
      HotV= #somewhat_Zero,
      ColdV= #somewhat_NB,
      Temp= #cold.
With Truth Degree of : 0.789931

shower_control_rule(Flow,Temp,ColdV,HotV) :-
      ColdV= #somewhat_Zero,
      HotV= #somewhat_NS,
      Flow= #high,
      Temp= #warm.
With Truth Degree of : 0.408403

shower_control_rule(Flow,Temp,ColdV,HotV) :-
      Flow= #strong,
      HotV= #somewhat_NS,
      Temp= #cool,
      ColdV= #somewhat_NM.
With Truth Degree of : 0.536515
```

```
shower_control_rule(Flow,Temp,ColdV,HotV) :-
      ColdV= #somewhat_Zero,
      HotV= #somewhat_NS,
      Temp= #hot,
      Flow= #fOK.
With Truth Degree of : 0.357865

shower_control_rule(Flow,Temp,ColdV,HotV) :-
      ColdV= #somewhat_Zero,
      HotV= #somewhat_NS,
      Flow= #high.
With Truth Degree of : 0.41344

shower_control_rule(Flow,Temp,ColdV,HotV) :-
      Flow= #strong,
      ColdV= #somewhat_NS,
      Temp= #warm,
      HotV= #somewhat_NM.
With Truth Degree of : 0.324645

shower_control_rule(Flow,Temp,ColdV,HotV) :-
      HotV= #somewhat_Zero,
      ColdV= #somewhat_NS,
      Flow= #strong.
With Truth Degree of : 0.239639

shower_control_rule(Flow,Temp,ColdV,HotV) :-
      Flow= #strong,
      HotV= #somewhat_NS,
      ColdV= #somewhat_Zero.
With Truth Degree of : 0.398868

shower_control_rule(Flow,Temp,ColdV,HotV) :-
      HotV= #somewhat_Zero,
      Temp= #cool,
      ColdV= #somewhat_NS.
With Truth Degree of : 0.175188

shower_control_rule(Flow,Temp,ColdV,HotV) :-
      HotV= #somewhat_Zero,
      ColdV= #somewhat_PS,
      Temp= #warm,
      Flow= #low.
With Truth Degree of : 0.365017

shower_control_rule(Flow,Temp,ColdV,HotV) :-
      HotV= #somewhat_Zero,
      ColdV= #somewhat_PS,
      Temp= #warm.
With Truth Degree of : 0.376894

shower_control_rule(Flow,Temp,ColdV,HotV) :-
      Flow= #strong,
      ColdV= #somewhat_NB,
```

Temp= #cool,
HotV= #somewhat_Zero.
With Truth Degree of : 0.535793

shower_control_rule(Flow,Temp,ColdV,HotV) :-
ColdV= #somewhat_Zero,
HotV= #somewhat_PS,
Temp= #cool.
With Truth Degree of : 0.260455

shower_control_rule(Flow,Temp,ColdV,HotV) :-
Flow= #strong,
ColdV= #somewhat_NB,
HotV= #somewhat_NS,
Temp= #cool.
With Truth Degree of : 0.444291

shower_control_rule(Flow,Temp,ColdV,HotV) :-
ColdV= #somewhat_NS,
HotV= #somewhat_NS,
Temp= #tOK,
Flow= #high.
With Truth Degree of : 0.644789

shower_control_rule(Flow,Temp,ColdV,HotV) :-
ColdV= #somewhat_NS,
Flow= #strong,
HotV= #somewhat_NM.
With Truth Degree of : 0.528823

shower_control_rule(Flow,Temp,ColdV,HotV) :-
HotV= #somewhat_NS,
ColdV= #somewhat_Zero,
Temp= #warm.
With Truth Degree of : 0.271843

shower_control_rule(Flow,Temp,ColdV,HotV) :-
HotV= #somewhat_Zero,
ColdV= #somewhat_NS.
With Truth Degree of : 0.178586

shower_control_rule(Flow,Temp,ColdV,HotV) :-
Flow= #strong,
ColdV= #somewhat_NM,
HotV= #somewhat_Zero,
Temp= #cool.
With Truth Degree of : 0.973207

shower_control_rule(Flow,Temp,ColdV,HotV) :-
ColdV= #somewhat_Zero,
HotV= #somewhat_NS,
Temp= #tOK.
With Truth Degree of : 0.288416

```
shower_control_rule(Flow,Temp,ColdV,HotV) :-
      Flow= #strong,
      HotV= #somewhat_NB,
      ColdV= #somewhat_NS,
      Temp= #warm.
With Truth Degree of : 0.378513

shower_control_rule(Flow,Temp,ColdV,HotV) :-
      HotV= #somewhat_NS,
      ColdV= #somewhat_NS,
      Temp= #tOK.
With Truth Degree of : 0.312828

shower_control_rule(Flow,Temp,ColdV,HotV) :-
      ColdV= #somewhat_Zero,
      Temp= #hot,
      HotV= #somewhat_NS.
With Truth Degree of : 0.289512

shower_control_rule(Flow,Temp,ColdV,HotV) :-
      Flow= #strong,
      Temp= #cold,
      ColdV= #somewhat_NB.
With Truth Degree of : 0.133221

shower_control_rule(Flow,Temp,ColdV,HotV) :-
      ColdV= #somewhat_Zero,
      HotV= #somewhat_NB,
      Temp= #hot.
With Truth Degree of : 0.268705

shower_control_rule(Flow,Temp,ColdV,HotV) :-
      Flow= #strong,
      ColdV= #somewhat_NM,
      HotV= #somewhat_NS.
With Truth Degree of : 0.682531

shower_control_rule(Flow,Temp,ColdV,HotV) :-
      ColdV= #somewhat_Zero,
      HotV= #somewhat_PS,
      Temp= #cold,
      Flow= #fOK.
With Truth Degree of : 0.454544

shower_control_rule(Flow,Temp,ColdV,HotV) :-
      HotV= #somewhat_Zero,
      ColdV= #somewhat_PS,
      Temp= #tOK,
      Flow= #little.
With Truth Degree of : 1.0

shower_control_rule(Flow,Temp,ColdV,HotV) :-
      Flow= #strong,
      HotV= #somewhat_NB,
```

```
    ColdV= #somewhat_Zero.
With Truth Degree of : 0.331212

shower_control_rule(Flow,Temp,ColdV,HotV) :-
    HotV= #somewhat_Zero,
    ColdV= #somewhat_PS,
    Temp= #hot.
With Truth Degree of : 0.0596341

shower_control_rule(Flow,Temp,ColdV,HotV) :-
    Flow= #strong,
    HotV= #somewhat_NM,
    ColdV= #'Zero',
    Temp= #hot.
With Truth Degree of : 0.836265
```