in *Machine Learning: A Multistrategy Approach, Vol. IV*, R.S. Michalski and G. Tecuci, (Eds.), Morgan Kaufmann Publishers, San Francisco, CA, 1994

16

LEARNING GRADED CONCEPT DESCRIPTIONS BY INTEGRATING SYMBOLIC AND SUBSYMBOLIC STRATEGIES

Jianping Zhang
(Utah State University)

Abstract

Concepts involved in real world applications usually possess graded structures. Instead of being equivalent, examples of a concept with a graded structure may be characterized by a degree of typicality in representing the concept. Both pure symbolic and subsymbolic representations are not adequate for describing concepts with graded structures. A pure symbolic representation fails to describe the graded structure of a concept, and a pure subsymbolic representation is unable to capture the central tendency of a concept. This chapter presents an integrated approach that combines a symbolic approach with a subsymbolic one to learn concepts with graded structures. Concepts in this approach are represented by a hybrid representation that is a combination of the symbolic and numeric representations. In the hybrid representation, the symbolic element explicitly captures the central tendency of a concept, and the numeric element implicitly handles the atypical aspect of a concept. A learning algorithm that adjusts both the symbolic and numeric elements of the representation to achieve the best fit between the description and the given concept examples is described. The method has been tested on several problems. The results have shown a statistically meaningful advantage to

the proposed method over the other methods on the problems that involved concepts with graded structures.

16.1 INTRODUCTION

Concepts involved in real world applications usually possess graded structure (Barsalou, 1985). Instead of being equivalent, examples of a concept may be characterized by a degree of typicality in representing the concept. The typicality of an example is determined by three factors: (1) the number of the common concept properties shared by the example, (2) the importance of the common concept properties shared by the example, and (3) the closeness by which the example matches the common concept properties. For example, a disease can be predicted by a set of symptoms. Patients of the disease may not possess all symptoms. Patients with more symptoms are more likely to have the disease than patients with fewer symptoms. All symptoms may not be equally important to predict the disease. The patients with important symptoms are more likely to have the disease than the patients with less important symptoms. Some symptoms, such as blood pressure, have a linear domain. The higher (or lower) blood pressure a patient has, the more likely the patient is to have the disease. A concept with a graded structure has a central tendency that is characterized by typical examples.

Pure symbolic representations, e.g., decision trees and rules, are not adequate for describing concepts with graded concepts because they represent a concept as a single symbolic description. In such a pure symbolic description, an example belongs to the concept only if it satisfies all conditions of the concept description. Otherwise, the example does not belong to the concept. Such a yes or no classification makes a pure symbolic representation impossible to capture the "graded" behavior of a concept. On the other hand, subsymbolic representations such as neural networks and Bayesian classifiers do not give explicit descriptions to represent the basic principles underlying a concept, so they fail to represent the central tendency of the concept. This chapter presents an integrated approach that combines a symbolic approach with a subsymbolic one to represent and learn concepts with graded structures.

There have been some efforts in combining these two approaches. Such efforts include the STAGGER system (Schlimmer, 1987), Perceptron trees (Utgoff, 1988), and the NGE approach (Salzberg, 1991). The differences between the approach reported in this chapter and the above approaches were discussed in Zhang and Michalski (1994). Towell and Shavlik (1994, Chapter 15 of this book) described an approach that combined symbolic representation with neural networks.

The approach presented employs a novel hybrid representation to describe concepts. The representation is a simple but powerful form of *two-tiered concept representation* (Michalski, 1987; Bergadano, et al., 1992) and combines symbolic

and numeric representations. The symbolic element of the representation explicitly describes the central tendency of a concept, and the numeric element extends the symbolic description to describe the atypical cases of the concept. A learning algorithm presented in this chapter adjusts both the symbolic and numeric elements of the representation to achieve the best fit between the description and the given concept examples.

The method has been implemented in the system FCLS (*F*lexible *C*oncept *L*earning *S*ystem) and tested on several problems. For comparison, other methods, such as the decision tree learning system C4.5 (Quinlan, 1987) and a rule learning system similar to AQ (Michalski and Larson, 1978), were tested on the same problems. The results have shown a statistically meaningful advantage to the proposed method over the other methods both in terms of the classification accuracy and the description simplicity on the problems that involve concepts with graded structure.

16.2 CONCEPT REPRESENTATION

In the proposed method, a concept is represented as a disjunction of Weighted Threshold Disjuncts (WTD), each of which consists of two elements: a symbolic element and a numeric element. A similarity measure unifies these two elements. The symbolic element of a WTD is a base disjunct that explicitly describes a central tendency of the concept. The numeric element consists of a set of weights and a threshold. Each condition of a base disjunct is associated with a weight that reflects the degree of necessity of the condition. The threshold defines the boundary of a WTD.

16.2.1 Weighted Threshold Disjunct (WTD)

A WTD is composed of a base disjunct that is a conjunction of conditions, a set of weights, and a threshold. In the rest of the chapter, we refer to a weighted threshold disjunct as a WTD. Base disjuncts are represented as complexes by the attribute based Logic System VL_1 (Michalski, 1983). A complex in VL_1 is a conjunction of conditions (formally called selectors), each of which is a relational expression

$$[L = R]$$

where L is an attribute, and R, called the *referent*, is a value or a disjunction of values from the domain of L. Such a condition is satisfied by an example if the value of the attribute L in this example is one of the values in R.

Each condition of a base disjunct has a weight that reflects the degree of necessity of the condition. Its value ranges from 0 to $+\infty$. The condition with weight 0 plays no role, while the condition weighted $+\infty$ is necessary. A larger weight means a more important condition.

In addition to weights, each WTD has a threshold that is a real number between 0 and 1. The threshold of a WTD defines the boundary of the WTD. An example is covered by a WTD if its similarity to the WTD is larger than or equal to the threshold of the WTD. The similarity of an example to a WTD is calculated by the similarity measure. Decreasing the threshold relaxes the requirements of the WTD that have to be satisfied to increase the coverage of the WTD.

16.2.2 Similarity Measure

The similarity measure (SM) measures the similarity between an example and a WTD. The SM of an example e and a WTD wtd is defined as the opposite of the distance measure D(e, wtd) normalized by the distance between the farthest example in the example space and wtd. Specifically, it is calculated as

$$SM(e,wtd) = 1 - \frac{D(e,wtd)}{MAX_{i=1..n}\{D(e_i, wtd)\}}$$

where e_1, ..., e_n are all examples in the example space. $D(e, wtd)$ is defined as a weighted sum of the distances between the example e and all conditions of wtd:

$$D(e, wtd) = \sum W_i * CD(e, C_i)$$

where W_i is the weight of the condition C_i. If W_i is $+\infty$, it needs to be specially calculated. $CD(e, C_i)$ is the distance between the example e and the condition C_i. The distance between an example and a condition depends on the type of the attribute involved in the condition. An attribute can be one of two types: nominal and linear. In a nominal condition, the referent is a single value or an internal disjunction of values; e.g., [color = red v blue v green]. The distance is 0 if such a condition is satisfied by an example and 1 otherwise. In a linear condition, the referent is a range of values or an internal disjunction of ranges; e.g., [weight = 1..3 v 6..9]. A satisfied condition returns the value of distance 0. If the condition is not satisfied, the distance between an example and the condition is the difference between the value of the example and the nearest end-point of the interval of the condition normalized by the distance between the farthest value and the condition. For example, if the domain of x is [0 .. 10], the value of x for the event e is 4, and the condition C is [x = 7 .. 9], then $CD(e, C) = \frac{7-4}{7-0} = \frac{3}{7}$.

When a condition with $+\infty$ weight is not satisfied, $D(e, wtd)$ is set to $MAX_{i=1..n}\{D(e_i, wtd)\}$ so that

$$SM(e, wtd) = 1 - \frac{(D(e, wtd)}{MAX_{i=1..n}\{D(e_i, wtd)\}} = 0$$

Also, $\infty * CD = 0$ if $CD = 0$.

16.2.3 Examples

To illustrate the idea of the hybrid representation, let us consider a simple example: learning the concept of a labor-management contract (Bergadano et. al, 1988). In this problem, the example space is divided, from the labor point of view, into acceptable and unacceptable contracts. Each contract is described by a set of selected attributes, e.g., general wage increase (gwi), job security (job-sec), extent of pension (p-ext), pension for overtime work, and fringe benefits. Let us assume that the class of acceptable contracts is represented by the following three disjuncts:

(job-sec = yes) & (p-ext = yes) or
(job-sec = yes) & (gwi = medium ∨ large) or
(p-ext = yes) & (gwi = medium ∨ large)

This description says that if a contract satisfies any two of the three conditions (job-sec = yes), (p-ext = yes), and (gwi = medium v large), the contract is acceptable. By using the hybrid representation, these three disjuncts merge into one WTD:

[job-sec = yes : 1] & [p-ext = yes : 1] & [gwi = medium ∨ large : 1]
Threshold = $\frac{2}{3}$ = 0.67

The number following a condition is the weight of the condition. The base disjunct includes three conditions:

[job-sec = yes] & [p-ext = yes] &[gwi = medium v large]

and represents the ideal situation of acceptable contracts. All three conditions are equally important. The description tells that contracts that satisfy all three conditions are ideal contracts, but those that only satisfy two of the three conditions are less ideal.

Now suppose acceptable contracts must have job security. The class of acceptable contracts is defined as the following WTD:

[job-sec = yes : ∞] & [p-ext = yes : 1] & [gwi = medium v large : 1]
Threshold = $\frac{1}{2}$ = 0.5

In this WTD, the condition [job-sec = yes] is necessary and must be satisfied by all acceptable contracts. The other two conditions are not necessary, and one of them must be satisfied.

Suppose that the attribute *gwi* is linear, and the order of its values is small, medium, and large. The WTD representing acceptable contracts becomes

[job-sec = yes : ∞] & [p-ext = yes : 1] & [gwi = large : 1]
Threshold = $\frac{1}{4}$ = 0.25

Under this WTD, the contract whose *gwi* is *large* is better than the contract whose *gwi* is *medium*. For example, let us consider the two contracts C1 (yes yes large) and C2 (yes yes medium). The similarity of C1 and C2 to the WTD is 1 and 3/4, respectively; so, C1 is better than C2.

16.3 CONCEPT RECOGNITION

Concept recognition in traditional concept representations, such as decision trees and rules, is very simple: if an example satisfies all conditions of a concept description, then it belongs to the concept; otherwise, it does not. This simple recognition method is called the strict method. This section describes a flexible classification method that was implemented in FCLS.

In the flexible method, an example is classified to a concept if it is uniquely covered by the concept description. An example is covered by a concept description if its similarity to one of the WTDs of the description is larger than or equal to the threshold of the WTD. The examples covered by no concept description are referred to as no-match examples, and the examples covered by more than one concept description are referred to as multiple match examples. In our method, the classification of no match and multiple match examples is determined by their Relative Similarity (RS) to WTDs. An example is classified as a concept if its RS to one of the WTDs of the concept description is the largest among all concept descriptions. RS between a WTD wtd and an example *e* covered by wtd is defined as follows:

$$RS(e, wtd) = \frac{SM(e, wtd) - th(wtd)}{1 - th(wtd)}$$

where *th*(*wtd*) is the threshold of wtd and smaller than 1. If *th*(*wtd*) = 1, *RS*(*e*, *wtd*) = 1. Because e is covered by wtd, *SM*(*e*,*wtd*) ≥ *th*(*wtd*). Thus, the RS of a multiple match example to a WTD that covers it is between 0 and 1.

The RS between a WTD wtd and an uncovered example *e* is computed as follows:

$$RS(e, wtd) = \frac{SM(e, wtd) - th(wtd)}{th(wtd)}$$

Because *e* is not covered by wtd, *SM*(*e*, *wtd*) < *th*(*wtd*) and RS of a no match example to a WTD is between 0 and –1. A covered example always has a larger RS than an uncovered example.

For each WTD, all training examples can be divided into three types, according to the way they are covered by the WTD. The examples in different types play different roles during learning.

1. *Strictly covered examples* are strictly covered by the base disjunct of the WTD; namely, they satisfy all conditions of the disjunct.
2. *Flexibly covered examples* are examples covered by the WTD.
3. *Nearly covered examples* are not flexibly covered, but their similarity to the WTD is close to the threshold. For a detailed definition of nearly covered examples, see Zhang (1990).

16.4 The Learning Algorithm

Table 16.1 shows the top-level learning algorithm. The input of the algorithm is a set of concepts with their examples. The output of the algorithm is a disjunction of WTDs for each given concept that covers all positive examples of the concept and no negative examples. This algorithm works in an iterative fashion. In each iteration, the concept description that has the largest error omission is generalized by generating a new WTD. The error omission of a concept description is the percentage of the number of the examples of the concept that are not covered by the description. The new WTD is generated to minimize the error omission.

Each iteration of the algorithm first generates a WTD; then the current concept descriptions are used to classify all examples using the flexible classification method. If all examples are correctly classified, the algorithm terminates and outputs the current descriptions; otherwise, it repeats.

16.4.2 The WTD Generation Algorithm

The WTD generation algorithm generates the "best" WTD for a given concept from a set of positive and negative examples. The process of generating the "best" WTD is divided into two phases. The first phase generates the symbolic

Table 16.1: The top-level learning algorithm

```
Let DES be empty
Repeat
    Select the concept CNPT that has the largest error omission
    Generalize CNPT by generating a WTD
    Add the new WTD into DES
Until All examples are correctly classified
Return DES
```

description, which is a set of consistent disjuncts. The algorithm of the first phase is called the Base Disjunct Generation Algorithm. The second phase optimizes the symbolic description by creating the numeric description, which contains the weights of all conditions and the threshold. The WTD Optimization Algorithm serves as the algorithm in the second phase.

16.4.2.1 Phase 1: The Base Disjuncts Generation Algorithm

The algorithm generates a set of the most general consistent disjuncts. Table 16.2 gives the disjunct generation algorithm. The algorithm starts with the most general disjunct that strictly covers the whole instance space. In order to find the consistent disjuncts, the covered negative examples must be excluded. The technique used in the algorithm is similar to the *star* algorithm of AQ (Michalski, 1983). During each cycle, the consistency of each disjunct in STAR is tested. If the disjunct is consistent, it is added to the set of CONSISTENT-DNTS and removed from STAR. Otherwise, it is specialized by removing an attribute value from one of its conditions. This specialization is repeated for each condition of the disjunct. The attribute value of a condition is chosen to maximize the number of negative examples and minimize the number of positive examples excluded from the disjunct. This yields several new disjuncts, each of which covers fewer negative examples. The new star is the union of these newly specialized disjuncts. A certain maximum number (MAXSTAR) of these disjuncts are selected for further processing. This set of disjuncts is selected based on their potential quality (Zhang, 1990). When STAR is empty, the algorithm terminates with a set of consistent disjuncts. Figure 16.1(a) shows the most general disjunct that the algorithm starts with, and Figure 16.1(b) shows the set of consistent disjuncts that the algorithm ends up with.

Table 16.2: Phase 1: The disjunct generation algorithm

```
Let STAR be the set containing the most general disjunct that covers all examples.
Let CONSISTENT-DNTS be empty.
Repeat
    Let NEWSTAR be empty
    For each disjunct DNT in STAR
        For each attribute
            select a value to remove so that a more specific disjunct NEWDNT is generated
            if NEWDNT is consistent,
                then add NEWDNT into CONSISTENT-DNTS
                else add NEWDNT into NEWSTAR
    Let STAR be MAXSTAR disjuncts with the largest potential quality in NEWSTAR.
until STAR is empty
Return CONSISTENT-DNTS
```

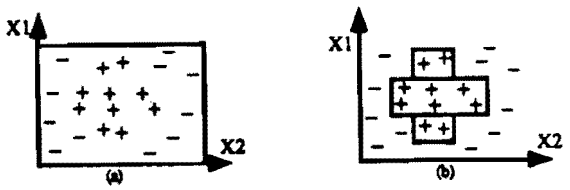

Figure 16.1: An illustration of the function of the phase 1. (a) The most general disjunct with which the algorithm starts. (b) The set of consistent disjuncts with which the algorithm ends up.

16.4.2.2 Phase 2: The WTD Optimization Algorithm

In a logic type representation, to match a disjunct, an example must satisfy all conditions of the disjunct. In the proposed representation, the conditions of a WTD that need to be satisfied by an example are controlled by the threshold of the WTD. The larger the threshold, the more conditions that need to be satisfied. Decreasing the threshold relaxes the conditions that have to be satisfied. The WTD optimization algorithm tries to increase the coverage of WTDs by decreasing their thresholds. In order to decrease the threshold of a WTD without increasing inconsistency, the similarity of nearly covered negative examples must be reduced so that the threshold can be decreased without covering more negative examples. One way to reduce the similarity of nearly covered negative examples is to specialize the base disjunct of the WTD by removing some values of conditions involved in the base disjunct. The value selected for specialization occurs on many nearly covered negative examples and few nearly covered positive examples. Thus, the process of optimizing a WTD is to further specialize its base disjunct and adjust its threshold.

The WTD optimization algorithm is similar to the base disjunct generation algorithm in that it also performs a general-to-specific beam search. In this algorithm, the threshold is decreased, and the base disjunct is specialized. Thus, a WTD is often generalized although its base disjunct is specialized. Another major difference involves the different negative examples that the two algorithms try to exclude. The base disjunct generalization algorithm reduces the number of strictly covered negative examples, whereas the WTD optimization algorithm reduces the number of nearly covered negative examples. This difference is reflected in the different potential quality evaluation functions and the different ways to select a value of a condition for specialization in the two algorithms. The potential quality of a disjunct is computed solely based on the strictly covered examples, but the potential quality of a WTD is computed based on the flexibly covered examples as well as the nearly covered examples. In the base disjunct generalization algorithm,

the value of a condition that occurs most frequently on strictly covered negative examples and least frequently on strictly covered positive examples is selected for specialization, but in the WTD optimization algorithm, the value of a condition that occurs most frequently on nearly covered negative examples and least frequently on nearly covered positive examples is selected for specialization. The quality evaluation functions will be discussed later.

Table 16.3 shows the WTD optimization algorithm. The algorithm first transfers the consistent disjuncts generated in phase 1 to WTDs by computing their weights and thresholds. The weight learning algorithm and the threshold adjustment algorithm will be introduced in the next section. The STAR is initialized as *MAXSTAR* of these WTDs with the highest potential quality. Then the "best" WTD is selected as the initial "best" WTD BEST-WTD. After the algorithm terminates, BEST-WTD is output. The "best" WTD is the WTD with the highest quality.

Like the base disjunct generation algorithm, the algorithm repeats the beam search until the stop condition is satisfied. In each cycle of the loop, a set of new WTDs is generated. Quality and potential quality are computed for each newly generated WTD. The WTD with the highest quality replaces BEST-WTD if its quality is larger than BEST-WTDs. The WTDs that have no value to be improved or cannot be improved are removed from NEWSTAR. Two kinds of WTDs—the most specific WTDs and the WTDs with 0 potential improvement—cannot be improved. The WTDs with very low quality (comparing with BEST-WTD) have no value to be improved. Issues about the potential improvement will be discussed later. The *MAXSTAR* new WTDs with the highest potential quality are selected for further improvement. *MAX-TRIES* is an integer parameter that controls the execution of the loop. If BEST-WTD has not been improved in *MAX-TRIES* steps, the algorithm stops.

Table 16.3: Phase 2: The WTD optimization algorithm

```
Repeat
    Let NEWSTAR be empty
    For each WTD in STAR
        For each attribute
            select a value to remove
            compute the weights and the threshold to generate a new WTD NEWWTD
            if NEWWTD has equal or higher quality than BEST-WTD
                then replace BEST-WTD by NEWWTD
                    set NO-IMPROVEMENT to 0
                else add 1 to NO-IMPROVEMENT
            add NEWWTD into NEWSTAR
    Remove all WTDs that cannot be improved from NEWSTAR
    Let STAR be MAXSTAR WTDs with the largest potential quality in NEWSTAR
until NO-IMPROVEMENT > MAX-TRIES or STAR is empty
Return BEST-WTD
```

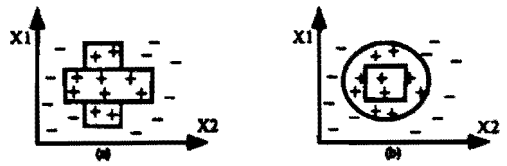

Figure 16.2: An illustration of the function of the phase 2. (a) Initial WTDs with which the algorithm starts. (b) The WTD with which the algorithm ends up.

Figure 16.2(a) shows the initial WTDs that the algorithm starts with, and Figure 16.2(b) shows the WTD that the algorithm ends up with. In Figure 16.2(b), the circle represents a WTD, and the square inside the circle represents its base disjunct. It can be seen that the disjunct in Figure 16.2(b) is more specific than the two disjuncts in Figure 16.2(a), but the WTD is more general than both of the two disjuncts in Figure 16.2(a).

16.4.3 Weight Learning and Threshold Adjustment

In the hybrid representation, each condition of a WTD is associated with a weight that is the degree of necessity of the condition. A weight is a real value ranging from 0 to $+\infty$. The larger the weight of a condition, the more necessary the condition. Weights are computed during learning. Each time a WTD is generated, its weights are computed. In computing the weight of a condition, the algorithm counts the number of positive and negative examples that do not match the condition

$$w(C) = \frac{p(\text{unmatched} \mid \text{NEG})}{p(\text{unmatched} \mid \text{POS})}$$

where $p(unmatched \mid NEG)$ and $p(unmatched \mid POS)$ are the fraction of negative and positive examples that do not match C. The range for $w(C)$ is from 0 to $+\infty$. When the condition C is satisfied by all positive examples, $p(unmatched \mid POS) = 0$ so that $w(C) = +\infty$, and the condition C is necessary. When the condition C is satisfied by all negative examples, $p(unmatched \mid NEG) = 0$ so that $w(C) = 0$. This case occurs seldom because such a condition is rarely generated by the learning algorithm. The fewer negative examples satisfy the condition C, the larger $p(unmatched \mid NEG)$ and $w(C)$. The more positive examples satisfy the condition C, the smaller $p(unmatched \mid POS)$, therefore the larger $w(C)$. When both $p(unmatched \mid NEG)$ and $p(unmatched \mid POS)$ are equal to 0, $w(C)$ is set to 1.

As described above, in phase 2, each time the base disjunct is specialized, the threshold is decreased to achieve the best fit between the WTD and given concept examples. The decrease in a threshold is divided into a number of steps; the thresh-

old is decreased by a fixed quantity D in each step. After each decrease, the WTD is evaluated to check if the WTD is improved by the decrease in the threshold. If it has not been improved in several steps, say, 3, then the adjustment of the threshold stops, and the threshold on which the WTD achieves the highest quality is the threshold of the WTD. The quantity D decreased in each step for the WTD wtd is determined as follows:

$$\Delta = \frac{1 - \mathrm{MIN}_{i=1,\ldots,n}\{SM(e_i, wtd)\}}{100}$$

where $e_1, \ldots, e_n$ are all training examples.

16.4.4 Concept Evaluation

Concept learning can be viewed as a search for the "best" solutions (concept descriptions). Because of the complexity of brute force search, all learning systems search the description space heuristically. Therefore, an evaluation function is needed as a heuristic to decide one or more intermediate hypothesis for further improvement. Such intermediate hypotheses should have potential for improvement. Also, an evaluation function is needed to determine the best description that is the final solution of the search when the search algorithm terminates. Most current inductive learning systems use one evaluation function for both purposes. It means that a better description always has greater potential for improvement. Unfortunately, this is not always true. In the following, some examples will be given to show that a good description may not have potential for improvement.

Three different evaluation functions are used in FCLS. The first one, called *Quality Evaluation Function*, evaluates the quality of a WTD; the second one, *Base Disjunct Potential Quality Evaluation Function*, evaluates the potential quality of a disjunct; and the third one, *WTD Potential Quality Evaluation Function*, evaluates the potential quality of a WTD. The WTD with the highest quality is the "best" WTD, but the WTD with the highest potential quality has the "greatest" potential for improvement. In the following subsections, these three evaluation functions are simply described; for further detail, see Zhang (1990).

16.4.4.1 The Quality Evaluation Function

As in most inductive learning systems, the quality of a WTD is evaluated based on its completeness and consistency with regard to the training examples. Generally, one can gain completeness at the expense of consistency, or one can gain consistency by sacrificing completeness. They are two competing criteria. For this reason, quality is evaluated as the product of these two parts: completeness, which is the percentage of positive examples covered, and consistency, which is the

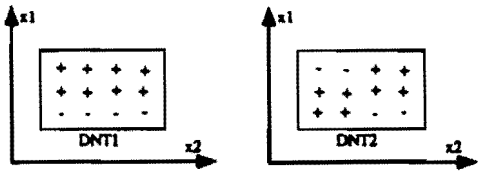

Figure 16.3: Illustration of the difference between the quality and potential improvement of disjuncts

ratio of the number of negative examples covered and the number of total examples covered.

16.4.4.2 Base Disjunct Potential Quality Evaluation Function

Before describing the details of the base disjunct potential quality evaluation function, let us first discuss some general issues about the potential quality. The potential quality is used to decide whether a WTD or a disjunct is worth being improved, whether a WTD or a disjunct can be improved, and how much a WTD or a disjunct can be improved. Based on the potential quality, both the base disjunct generalization algorithm and the WTD optimization algorithm determine which disjunct or WTD should be selected for further improvement. The potential quality consists of two parts: current quality and potential improvement. Current quality is similar to the quality and is computed based on the completeness and consistency of a WTD or a disjunct, but the concern of potential improvement is how much improvement can be achieved on the basis of current quality. The WTD (disjunct) with low current quality is not worth being improved even it has high potential improvement, but the WTD (disjunct) with low potential improvement has little potential for further improvement.

In evaluating the potential quality of a disjunct, only the examples strictly covered by the disjunct are considered. The current quality of a disjunct is computed based on the number of positive and negative examples strictly covered by the disjunct. The potential improvement of a disjunct is computed based on the distribution of the positive and negative examples inside the disjunct. Some distributions make a disjunct much easier to improve than the others. For example, Figure 16.3 shows two disjuncts DNT1 and DNT2 that cover the same number of positive and negative examples and have the same current quality. However, DNT1 is much easier to improve than DNT2 because the positive examples covered by DNT2 are scattered, but the positive examples covered by DNT1 are very concentrated. A disjunct with dispersed covered positive examples is hard to improve, so it has low potential improvement. A disjunct with concentrated covered positive examples is relatively easier to specialize to a consistent disjunct without losing much coverage of positive examples, so it has high potential improvement.

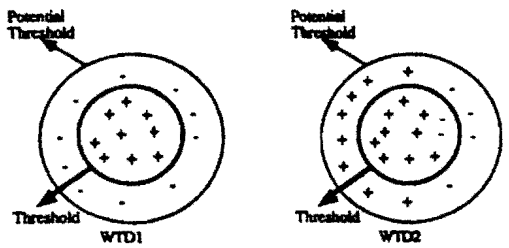

Figure 16.4: Illustration of the difference between the current quality and potential improvement of WTDs in phase 2

16.4.4.3 WTD Potential Quality Evaluation Function

The WTD potential quality evaluation function evaluates the potential quality of a WTD and is used in the WTD optimization algorithm. This algorithm tries to increase the coverage of positive examples of WTDs by decreasing their thresholds. The nearly covered examples most probably become covered after the threshold decreases. The more nearly covered positive examples a WTD has, the higher the potential improvement. If all nearly covered examples are negative, the WTD is surrounded by negative examples, and the decrease of its threshold can only increase the coverage of negative examples. Such a WTD cannot be improved and has no potential improvement.

In Figure 16.4, inside circles are thresholds of the two WTDs, and the outside circles are the potential thresholds of the two WTDs. The potential threshold of a WTD is used to decide nearly covered examples. The way to decide a potential threshold was described in Zhang (1990). All examples between a threshold and a potential threshold are nearly covered examples. The symbols + and – represent positive and negative examples, respectively. In this figure, WTD1 has higher current quality than WTD2 but has little potential improvement because it is surrounded by negative examples. In contrast, WTD2 has lower current quality but has larger potential improvement.

The potential improvement and the current quality of a WTD are computed based on the nearly covered examples and the examples covered by the WTD, respectively. The more positive nearly covered examples, the higher the potential improvement. Each of the nearly covered examples is associated with a value that ranges from 0.5 to 1. This value is used in evaluating the potential improvement. Examples closer to the threshold get larger value than examples farther from the threshold because the nearly covered examples closer to the threshold have more chance to become covered examples than the examples farther from the threshold. The value of all covered examples can be considered as 1. Thus, most of the nearly covered examples get a smaller value than covered examples. This is reasonable

Table 16.4: Positive and negative examples of the acceptable labor contracts

#	job-sec	p-ext	gwi	p-ovt	Class
1	yes	yes	large	yes	positive
2	yes	yes	small	no	positive
3	no	yes	medium	no	positive
4	yes	no	large	yes	positive
5	yes	yes	small	yes	positive
6	no	yes	medium	no	positive
7	no	no	small	no	negative
8	yes	no	small	no	negative
9	no	yes	small	yes	negative
10	no	no	large	yes	negative
11	yes	no	small	yes	negative
12	no	yes	small	no	negative

because nearly covered examples only have a chance to become covered, but they are not covered yet.

16.4.5 An Example Illustrating the Learning Algorithm

Let us go back to the labor contract example again. In addition to the three attributes—job-sec, p-ext, and gwi—a less important attribute, p-ovt (pension for overtime work), is added to the problem. Each object in the domain is now described by four attributes. Table 16.4 shows all examples given. The examples 1 to 6 are positive examples, and the examples 7 to 12 are negative examples.

We will ignore the weight learning and set *MAXSTAR* to 1 in the following example. First, the top level algorithm calls the WTD generation algorithm. The base disjunct generation algorithm starts with the most general disjunct:

[job-sec = yes ∨ no][p-ext = yes ∨ no][gwi=large ∨ medium ∨ small][p-ovt = yes ∨ no]

For each of the four conditions, the algorithm chooses one value to remove and generates four new, more specific disjuncts. For example, it chooses the value *no* to remove from the first condition because removing the value *no* excludes more negative examples (7, 9, 10, and 12) and fewer positive examples (3 and 6) from the disjunct than removing the value *yes*. This yields the following four new disjuncts:

[job-sec = yes][p-ext = yes ∨ no][gwi = large ∨ medium ∨ small][p-ovt = yes ∨ no]

[job-sec = yes ∨ no][p-ext = yes][gwi = large ∨ medium ∨ small][p-ovt = yes ∨ no]

[job-sec = yes ∨ no][p-ext = yes ∨ no][gwi = large ∨ medium][p-ovt = yes ∨ no]

[job-sec = yes ∨ no][p-ext = yes ∨ no][gwi = large ∨ medium ∨ small][p-ovt = yes]

One (because *MAXSTAR* = 1) of the four disjuncts is selected based on their potential quality for further improvement. For simplicity, we will choose the most consistent disjunct:

[job-sec = yes ∨ no][p-ext = yes ∨ no][gwi = large ∨ medium][p-ovt = yes ∨ no]

which covers 4 positive examples and one negative example, and has higher consistency than the other three disjuncts, so it is chosen for further improvement. Repeat the same process, and the following four new disjuncts are generated:

[job-sec = yes][p-ext = yes ∨ no][gwi = large ∨ medium][p-ovt = yes ∨ no]

[job-sec = yes ∨ no][p-ext = yes][gwi = large ∨ medium][p-ovt = yes ∨ no]

[job-sec = yes ∨ no][p-ext = yes ∨ no][gwi = medium][p-ovt = yes ∨ no]

[job-sec = yes ∨ no][p-ext = yes ∨ no][gwi = large ∨ medium][p-ovt = no]

All these four disjuncts cover no negative example, but they cover a different number of positive examples. The second disjunct that covers the largest number of positive examples is chosen as the output of the base disjunct generation algorithm. The disjunct

[job-sec = yes ∨ no][p-ext = yes][gwi = large ∨ medium][p-ovt = yes ∨ no]
Threshold = 1

serves as the base disjunct of the initial WTD for the WTD optimization algorithm. Actually, the WTD can be rewritten as

[p-ext = yes][gwi = large ∨ medium]
Threshold = 1

because the conditions [SHAPE = round ∨ square] and [COLOR = white ∨ black] include all values.

In the second phase, this WTD is optimized by specializing its base disjunct and decreasing its threshold. The way to specialize its base disjunct is the same as in phase 1, namely, removing a value from a condition. Because no value can be removed from the condition [p-ext = yes], only three new WTDs are generated. The following WTD is the best one of the three new WTDs:

[job-sec = yes][p-ext = yes][gwi = large ∨ medium][p-ovt = white ∨ black]
Threshold = 2/3

This is same as the WTD

[job-sec = yes][p-ext = yes][gwi = large ∨ medium]
Threshold = 2/3

This WTD covers all six positive examples and none of the negative examples.

16.5 EMPIRICAL EVALUATION

To evaluate the approach described in this chapter, a number of experiments were conducted on various domains with FCLS and C4.5. This section first outlines the experimental methods, then reports the experimental results.

16.5.1 Experimental Design

To thoroughly evaluate the approach, FCLS was tested on eight problems. This chapter will report the results from three problems: M-Of-N concept, congressional voting records, and lymphatic cancer. The results from other problems are reported in Zhang and Michalski (1994).

The learning methods involved in the experiments were a rule learning algorithm, a decision tree learning algorithm, and the algorithm reported in this chapter with and without weight learning. The rule learning algorithm was the base disjunct generation algorithm described; it generated a disjunction of disjuncts as a concept description that was equivalent to the description generated by AQ11 (Michalski and Larson, 1978). It provided the performance baseline for other methods. The decision tree learning algorithm was C4.5 (Quinlan, 1987).

The performance was evaluated on two aspects: classification accuracy and description complexity. Classification accuracy was measured as the percentage of correct classifications made by the concept description on a set of randomly selected test events. Description complexity was measured by the number of WTDs (disjuncts in rule learning) involved in a description. The complexity of decision trees is measured by the number of leaves in a tree.

In all experiments, FCLS ran on randomly generated training sets of various sizes. For each training set size, FCLS was run on four different randomly generated training sets. The final descriptions produced from these four runs were tested for accuracy on a set of test events and were measured for complexity. The results reported are the average of the four runs.

16.5.2 Experimental Results

The experiments described in this section were performed on three problems: M-Of-N concept, congressional voting records, and lymphatic cancer.

The problem of the M-Of-N concept contains two classes (positive and negative) and 10 nominal attributes, each of which has four values (0, 1, 2, and 3). The

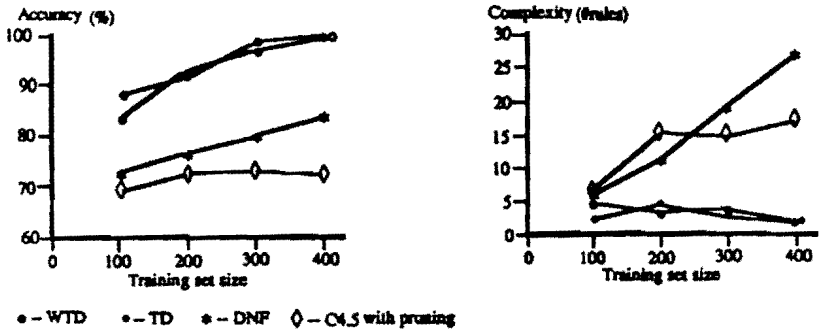

Figure 16.5: Classification accuracy and description complexity on M-Of-N concept

rule for distinguishing positive class from negative has the general form of "at least M of N conditions are satisfied." Specifically, the rule is "if any 5 or more of the first 7 attributes of an event are equal to 0 or 1, then the event belongs to the positive class; otherwise, it belongs to the negative class." The test set included 1,000 randomly selected events and was disjoint with training sets.

Figure 16.5 shows the results of the experiments from M-Of-N concept. In the figure, DNF is the rule learning method that generates DNF expression as a concept description. TD (Threshold Disjunct) represents the method described in the chapter with threshold adjusting but not weight learning. WTD represents the method reported in the chapter with both threshold adjusting and weight learning.

Significant improvements were achieved on both accuracy and complexity by the TD and WTD methods over the DNF method and C4.5 with all training set sizes. The TD and WTD methods have very similar performance. This is because all conditions of the target concept description were equally important, and weights play no role. Actually, the accuracy of the TD method was slightly higher than that of the WTD method, which can be explained as follows. In the WTD method, weights were adjusted based on the training examples during learning. Although all weights learned were close to each other, they were not exactly equal, so that the final description did not exactly describe the target concept that had the same weights for all conditions. The TD method produced the exact target concept descriptions for both the positive and negative classes at three of four training sets of 300 examples and all four training sets of 400 examples. The WTD method generated the target concept with slightly different weights. The significant performance improvement on this problem can be attributed to the clear graded structure that the M-Of-N concept possesses.

The data regarding the U.S. congressional voting records represented the 1981 voting records of 100 selected representatives, each of which was characterized by 19 binary attributes. Each attribute represented a vote and had two values: vote for and vote against. The problem was to learn descriptions discriminating

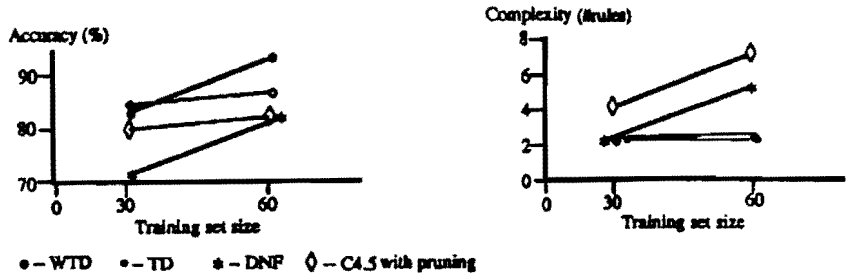

Figure 16.6: Classification accuracy and description complexity on congressional voting records

between the voting records of Democrats and Republicans. The two concepts involved in this problem, voting records of Democrats and voting records of Republicans, had graded structures. For example, Democrats had a tendency to vote for some issues and vote against others. Four training sets for each of two training sizes (30, 60) were formed by randomly drawing examples from the 100 examples; the other 40 examples were used as the testing set. The results are reported in Figure 16.6. The WTD method improved the accuracy over the DNF method and outperformed the TD method at the size of 60. The TD method achieved better results than the DNF method and performed slightly better than the WTD method at the size of 30. At the size of 60, the WTD and TD methods obtained simpler descriptions than the DNF method.

The lymphatic cancer data were characterized by 18 attributes and 4 diagnostic classes. Data of 148 patients were available. Twenty-five, 50, 75, and 100 examples were randomly selected for learning; the remaining 48 examples were used as a testing set. The results are shown in Figure 16.7. Both the WTD and TD methods gave better accuracy, except the size of 25, than the DNT method, but the improvement is not significant. The description generated by the WTD and TD methods was simpler than that given by the other two methods. A striking result was that all

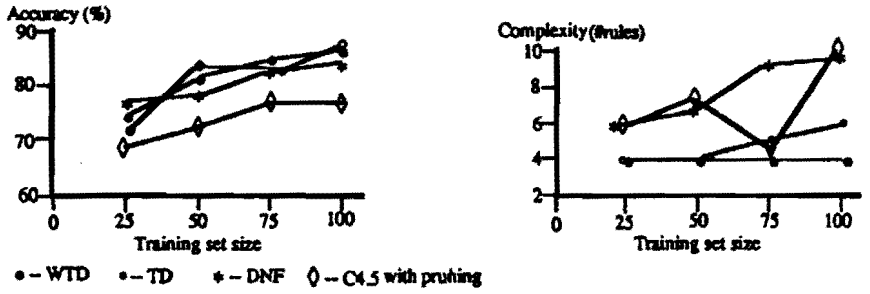

Figure 16.7: Classification accuracy and description complexity on lymphatic cancer

concept descriptions generated by the WTD method consist of only one WTD, with more conditions involved in the base disjunct. It seems that these base disjuncts captured the central tendencies of these four diagnostic classes.

16.6 SUMMARY AND FUTURE WORK

This chapter described a novel integrated approach to learning concepts with graded structures. In this approach, a hybrid representation that combines symbolic and numeric representations was proposed to explicitly describe the central tendency of a concept and implicitly extend the meaning of a concept by a threshold, a set of weights, and a similarity measure to handle the atypical aspect. A two step learning algorithm was designed and implemented to automatically acquire both symbolic and numeric descriptions from given examples. The method was tested in five domains, and the results were very promising and encouraging.

A number of problems need to be addressed in the future. First, the syntactic similarity measure needs to be augmented with a knowledge-based semantic similarity measure. This semantic similarity measure will include a set of inference rules and defines the similarity between an event and a WTD based on the semantics. Another related future research topic is constructive induction. In general, constructive induction produces descriptions that are easier to understand and capture the salient features of concepts. It would be useful to apply constructive induction to generate disjuncts that capture the principles of concepts.

Second, the problem of incremental learning should be addressed. The distribution of knowledge between the symbolic and numeric description may need to be adjusted when a set of new examples becomes available. Thus, incremental learning in the hybrid representation involves not only creating new descriptions but also adjusting the distribution between the symbolic and numeric description.

Finally, the typicality of examples could be very helpful in learning graded concept descriptions. Human experts are often very good at providing such information by telling which examples are typical and which are not. The typicality of examples can be used to reduce the search space and find the optimum distribution of concept meaning between the symbolic description and the numerical one. In the future, we will develop an algorithm to take advantage of typicality of examples in learning concept descriptions. In the algorithm, typical examples will be used first to learn disjuncts, and then less typical examples will be used to extend the disjuncts to WTDs.

ACKNOWLEDGMENTS

This research was done while the author was with the Artificial Intelligence Center of George Mason University. The author thanks R.S. Michalski and G. Tecuci for many discussions on the work reported in this chapter. This research was

done in the Artificial Intelligence Center of George Mason University. The research activities of the Center are supported in part by the Office of Naval Research under Grants No. N00014-88-K-0397, No. N00014-88-K-0226, No. N00014-90-J-4059, and No. N00014-91-J-1351; in part by the National Science Foundation under the Grant No. IRI-9020266; and in part by the Defense Advanced Research Projects Agency under the grants administered by the Office of Naval Research, No. N00014-87-K-0874 and No. N00014-91-J-1854. The author was also partially supported by a Utah State University Faculty Grant.

References

Barsalou, L., "Ideals, Central Tendency, and Frequency of Instantiation as Determinants of Graded Structure in Categories," *Journal of Experimental Psychology: Learning, Memory and Cognition*, 11, pp. 629–654, 1985.

Bergadano, F., Matwin, S., Michalski, R.S., and Zhang, J., "Representing and Acquiring Flexible Concepts in Knowledge-based Systems," In *Proceedings of the 3rd International Symposium on Methodologies for Intelligent Systems*, Torino, Itali, October, 1988.

———, "Learning Two-tiered Descriptions of Flexible Concepts," in *Machine Learning*, Vol. 8, No. 1, pp. 5–43, 1992.

Michalski, R.S., "A Theory and Methodology of Inductive Learning," In *Machine Learning: An Artificial Intelligence Approach*, 83–134, Michalski, Carbonell, and Mitchell (eds.), Morgan Kaufmann, San Mateo, CA, 1983.

———, "How to Learn Imprecise Concept: A Method Employing a Two-tiered Representation for Learning," In *Proceedings of the Fourth International Workshop on Machine Learning*, 50–58, (ed.) Pat Langley, Morgan Kaufmann, San Mateo, CA, 1987.

———, "Learning Flexible Concepts: Fundamental Ideas and a Method Bases on Two-tiered Representation," In *Machine Learning: An Artificial Intelligence Approach*, 63–111, Volume III, Kodratoff and Michalski (eds.), Morgan Kaufmann, San Mateo, CA, 1990.

Michalski, R.S. and Larson, J.B., "Selection of Most Representative Training Examples and Incremental Generation of VI1 Hypotheses: The Underlying Methodology and the Description of Programs ESEL and AQ11," Technical Report 867, Department of Computer Science, University of Illinois, Urbana, IL, 1978.

Quinlan, J.R., "Simplifying Decision Trees," In *International Journal of Man-Machine Studies*, Vol. 27, pp. 221–234, 1987.

Salzberg, S., "A Nearest Hyperrectangle Learning Method," *Machine Learning*, 6, pp. 251–276, 1991.

Schlimmer, J.C., *Concept Acquisition through Representational Adjustment*, Ph.D. thesis, Department of Information and Computer Science, University of California, Irvine, 1987.

Towell J.G, and Shavlik, J.W., "Refining Symbolic Knowledge Using Neural Networks," In *Machine Learning: A Multistrategy Approach*, Vol. IV, R.S Michalski and G. Tecuci (Eds.), Morgan Kaufmann Publishers, San Mateo, CA, 1994.

Utgoff, P.E., "Perceptron Trees: A Case Study in Hybrid Concept Representations," In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pp. 601–606, AAAI Press, Menlo Park, CA, 1988.

Zhang, J., "Learning Flexible Concepts from Examples: Employing the Ideas of Two-Tiered Concept Representation," Ph.D. Thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, 1990.

Zhang, J. and Michalski, R.S., "Combining Symbolic and Numeric Representations in Learning Flexible Concepts: The FCLS System," Submitted to *Machine Learning*, 1994, forthcoming.