Complementary Discrimination Learning: A Duality between Generalization and Discrimination

Wei-Min Shen
ACT/AI, MCC, 3500 West Balcones Center Drive
Austin, TX 78759
wshen@mcc.com

Abstract

Although generalization and discrimination are commonly used together in machine learning, little has been understood about how these two methods are intrinsically related. This paper describes the idea of *complementary discrimination*, which exploits semantically the syntactic duality between the two approaches: discriminating a concept is equivalent to generalizing the complement of the concept, and vice versa. This relation brings together naturally generalization and discrimination so that learning programs may utilize freely the advantages of both approaches, such as learning by analogy and learning from mistakes. We will give a detailed description of the complementary discrimination learning (CDL) algorithm and extend the previous results by considering the effect of noise and analyzing the complexity of the algorithm. CDL's performance on both perfect and noisy data and its ability to manage the tradeoff between simplicity and accuracy of concepts have provided some evidence that complementary discrimination is a useful and intrinsic relation between generalization and discrimination.

Introduction

Learning by generalization and learning by discrimination are two basic approaches commonly used in machine learning. In learning by generalization, the search for a target concept proceeds from specific to general guided by the similarity between instances. Winston's work (1975) on arch learning is a typical example. In learning by discrimination, on the other hand, the search proceeds from general to specific guided by the difference between instances. Feigenbaum and Simon's EPAM (1984) and Quinlan's ID3 (1983) serve as good representatives. Although much work, such as AM (Lenat 1977), Version Spaces (Mitchell 1982), Counterfactuals (Vere 1980), and STAR (Michalski 1983) have used both methods, little of it has revealed the relation between these two seemingly very different approaches.

This paper describes the idea of *complementary discrimination* (Shen 1989). The key observation is that discriminating a concept is equivalent to generalizing its complement, and vice versa. This is to say that the effect of discrimination and generalization can be achieved either way. For example, generalizing a concept using the similarity between instances can be accomplished by discriminating the concept's complement using the difference between instances, and vice versa. Exploiting this duality brings together naturally both discrimination and generalization so that a learning algorithm can make intelligent choices about which approach to take. For example, if a task requires learning from mistakes, then one might prefer using discrimination to achieve generalization. On the other hand, if there are existing theories for finding the relevant similarities between instances, then using generalization to accomplish discrimination might be better. The CDL algorithm to be described here has implemented only the part that uses discrimination to achieve generalization, yet it has already demonstrated some very encouraging results.

CDL learns concepts from training instances. It can incrementally learn any subset of the instance space (or concepts) from either perfect or noisy data, although some concepts may be preferred over others. Since CDL uses Predicate Calculus to represent concepts, it can deal with variables and relations, and can be easily integrated with problem solving systems and autonomous learning systems. Although CDL's learning is supervised, when combined with problem solving and experimentation its behavior is autonomous.

In some earlier papers (Shen 1989, Shen and Simon 1989), we have shown how CDL is used for learning from the environment and compared it with Version Spaces, STABB and discovery systems like GLAUBER and DALTON. In this paper, we will give a detailed description of the algorithm and report its performance on two typical concept formation tasks to show how CDL learns from noisy data and data that has exceptional cases. We will also evaluate CDL's complexity and analyze the quality of its learning.

The CDL Algorithm

CDL is an algorithm that learns concepts from training instances. It learns a concept by learning, in parallel, the concept itself, C, and its complement, $\overline{C}$. Driven by the data, CDL will incrementally revise the complemental form, either C or $\overline{C}$, whose prediction about a new instance is incorrect. It will first find the differences between the new instance and the previous instances that are correctly classified, then discriminate the complemental form by conjoining it with the difference. After that, CDL will update, or generalize, the complement of the complemental form to be the negation of the new complemental form.

This paper uses only propositional logic but the algorithm still works with first order logic. Here, the representation of instances is binary-valued feature vectors and the representation of concepts is propositional logical formulas. Multiple-valued features will be represented by multiple feature variables. For example, if a feature F has values a, b and c, we will use two variables x and y so that F=a will be $xy = 00$, F=b will be $xy = 01$ and F=c will be $xy = 10$. Although features are binary-valued, the representation allows "don't care" (*) as a value of features so that features with unknown values can be represented. Multiple classes (or concepts) will be represented by class variables. For example, to learn multiple classes W, X, Y and Z, we will use two class variables c_1 and c_2 so that W, X, Y and Z will be $c_1c_2 = 00$, $c_1c_2 = 01$, $c_1c_2 = 10$ and $c_1c_2 = 11$ respectively. These class variables will be given as inputs to the learning algorithm as if they are some extra features of instances. Thus, instead of telling CDL that an instance I belongs to a class C, we will state that $(I\ C)$ is a positive instance. For example, to say that an instance (1 0 0 1) belongs to W, we will say that $(1\ 0\ 0\ 1\ \overline{c}_1\ \overline{c}_2)$ is a positive instance; and to say that an instance (1 1 0 0) does not belong to Y, we will say $(1\ 1\ 0\ 0\ c_1\ \overline{c}_2)$ is a negative instance. Using these class variables, CDL can learn multiple concepts even though syntactically it learns only a single concept. For example, W, X, Y and Z will be learned as C = $(\cdots$ W) $\vee$ $(\cdots$ X) $\vee$ $(\cdots$ Y) $\vee$ $(\cdots$ Z).

Concepts will be logical propositions, such as $x \wedge y$ or $\overline{x} \vee \overline{y}$. The complement of a concept will be the negation of the concept. For example, $x \wedge y$ and $\overline{x} \vee \overline{y}$ are complements to each other. For convenience, concepts will be either in DNF or CNF; if a concept is in DNF then its complement will be in CNF, and vice versa. To classify a new instance, a concept will be matched to the instance. For example, if the instances are vectors of $(a\ b\ c)$ then a concept $a \wedge b$ will match $(1\ *\ 0)$ but not $(*\ 0\ *)$. As extremes, a concept T will match everything and a concept NIL will match nothing.

Table 1 shows the core of the algorithm. The procedure IN-CLASS?(i) serves as the supervisor; it returns T when the instance i is positive, or NIL when i is negative. The procedure PREDICT(i,C) returns T if i is matched by C and NIL otherwise (matched by $\overline{C}$). When a new instance comes, CDL makes a prediction by calling PREDICT on the instance. If the prediction equals the value of IN-CLASS?(i), the concepts remain the same. Otherwise, CDL will call DIFFERENCES to find the differences between this troublesome instance and previous classified instances, then use REVISE to update the complemental form, either C or $\overline{C}$, that made the incorrect prediction by conjoining it with the differences. The complement of the complemental form is then set to be the negation of the revised complemental form.

```
Let C be a concept and C̄ be C's complement.
Let POS be a set of positive examples.
Let NEG be a set of negative examples.

Procedure CDL()
  Let C be T and C̄ be NIL;
  While i ← GET-THE-NEXT-INSTANCE()
   If PREDICT(i,C) ≠ IN-CLASS?(i)
   then if PREDICT(i,C)=T (i.e. C covers too much)
        then D ← DIFFERENCES(i,POS)
             C ← REVISE(C,D,NEG)
             C̄ ← ¬C
        else (i.e. C̄ covers too much)
             D ← DIFFERENCES(i,NEG)
             C̄ ← REVISE(C̄,D,POS)
             C ← ¬(C̄);
  If IN-CLASS?(i)=T
    then insert(i,POS) else insert(i,NEG).

Procedure DIFFERENCES(i,INSTS)
  Let DIFF be { j\i | j ∈ INSTS} where j\i is the set
     of features that are in j but not in i;
  Return the minimum subset of DIFF
     that covers M% of INSTS.

Procedure REVISE(X,D,OPPOSITE)
  If X=T then return D else Y ← X∧D;
  For each y in Y
    If y is logically subsumed by some z∈Y
    then delete y from Y;
    If y = y'∧a ∧ ¬a (i.e. y contains contradictions)
    then Let P="covers no instance in OPPOSITE",
      If y'∧a satisfies P and y'∧¬a does not,
      then replace y by y'∧a;
      If y'∧¬a satisfies P and y'∧a does not,
      then replace y by y'∧¬a else delete y from Y;
  Return Y.
```

Table 1: The core of the CDL learning algorithm.

To illustrate the procedure DIFFERENCES(i,INSTS), consider i = $(a\overline{b}cd)$ and INSTS = $\{(\overline{a}\overline{b}\overline{c}d)\ (\overline{a}\overline{b}cd)\ (\overline{a}b\overline{c}d)\ (abc\overline{d})\}$. Since $(\overline{a}\overline{b}\overline{c}d) \setminus (a\overline{b}cd) = (\overline{a}\overline{c})$, $(\overline{a}\overline{b}cd) \setminus (a\overline{b}cd) = (\overline{a})$, $(\overline{a}b\overline{c}d) \setminus (a\overline{b}cd) = (\overline{a}b\overline{c})$ and $(abc\overline{d}) \setminus (a\overline{b}cd) = (b\overline{d})$, DIFF will be $\{(\overline{a}\overline{d})(\overline{a})(\overline{a}b\overline{c})(b\overline{d})\}$. Among the four elements in DIFF, the procedure will return $\{(\overline{a})\ (b\ \overline{d})\}$ because this subset is the minimum that covers all the

Description				Representation			
Class	height	hair	e yes	a	b	c	d
−	short	blond	brown	0	0	1	0
−	tall	dark	brown	1	0	0	0
+	tall	blond	blue	1	0	1	1
−	tall	dark	blue	1	0	0	1
−	short	dark	blue	0	0	0	1
+	tall	red	blue	1	1	0	1
−	tall	blond	brown	1	0	1	0
+	short	blond	blue	0	0	1	1

Table 2: A simple learning task.

instances in INSTS. For all the experiments in this paper, the parameter M is set to 90. In a later section, however, we will suggest a way to determine M automatically. The result returned by this procedure is always interpreted as a DNF logical form; so $\{(\overline{a})\ (b\ \overline{d})\}$ means $\overline{a} \vee b\overline{d}$.

The main task of procedure REVISE(X, D, OPPOSITE) is to conjoin the differences D with the concept X and simplify the results. In case when the results contain contradictions, the procedure also decides how to resolve the contradictions. For example, consider D $= \overline{a} \vee b\overline{d}$, X $= \overline{a}b \vee cd$, and OPPOSITE $= \{(abc\overline{d})(ab\overline{c}\,\overline{d})(a\overline{b}c\overline{d})\}$. Then Y will be X$\wedge$D $= \overline{a}b \vee \overline{a}cd \vee \overline{a}b\overline{d} \vee bcd\overline{d}$. Among these literals, $\overline{a}b\overline{d}$ will be deleted because it is subsumed by $\overline{a}b$; $bcd\overline{d}$ contains a contradiction and will be replaced by bcd because bcd does not cover any instance in OPPOSITE while $bc\overline{d}$ does.

Let us now observe how CDL performs on the learning task from (Quinlan 1983). Each instance is described as a conjunction of three attribute-value pairs, using the attributes: *height*, color of *hair*, and color of *eyes*. The instances and their representation in CDL are shown in the Table 2. We use variable a to represent height (0 for short and 1 for tall), bc for color of hair (00 for dark, 01 for blond and 10 for red), and d for color of eyes (0 for brown and 1 for blue).

The performance of CDL on this task is summarized in Table 3. When the first instance arrives, CDL predicts that the instance is positive (C matches it), but the prediction is wrong. Since there is no previous positive instance, procedure DIFFERENCES returns NIL and procedure REVISE sets C to NIL and $\overline{C}$ to T. On the second instance, CDL predicts that it is negative ($\overline{C}$ matches it). Since the prediction is correct, the concepts remain the same. On the third instance, CDL's prediction is negative but wrong. This time the procedure DIFFERENCES finds the differences between (1 0 1 1) and previous negative examples {(1 0 0 0) (0 0 1 0)} to be $\overline{c}\,\overline{d} \vee \overline{a}\,\overline{d}$. The procedure REVISE sets $\overline{C}$ to be equal to the difference $\overline{c}\,\overline{d} \vee \overline{a}\,\overline{d}$ (because the current $\overline{C}$ is T) and sets C to be the new $\overline{C}$'s complement: $(c \vee d)(a \vee d)$.

Based on the new concepts, CDL predicts that the fourth instance is positive but the prediction is wrong. This time, the instance (1 0 0 1) is compared with previous positive examples {(1 0 1 1)} and the differences found is (c). This difference is conjoined with C to form the new concept C, $(a \vee d)(c)$, and $\overline{C}$ is set to be the new C's complement: $\overline{a}\,\overline{d} \vee \overline{c}$.

Insts	Pred	Class	Diff	C	$\overline{C}$
				T	NIL
0010	T	NIL	NIL	NIL	T
1000	NIL	NIL			
1011	NIL	T	$\overline{a}\,\overline{d} \vee \overline{c}\,\overline{d}$		$\overline{a}\,\overline{d} \vee \overline{c}\,\overline{d}$
				$(a \vee d)(c \vee d)$	
1001	T	NIL	c	$(a \vee d)(c)$	
					$\overline{a}\,\overline{d} \vee \overline{c}$
0001	NIL	NIL			
1101	NIL	T	$\overline{b}$		$\overline{b}\,\overline{a}\,\overline{d} \vee \overline{b}\,\overline{c}$
				$(b \vee a \vee d)(b \vee c)$	
1010	T	NIL	d	$(b \vee c)(d)$	
					$\overline{b}\,\overline{c} \vee \overline{d}$
0011	T	T			

Table 3: A performance on the learning task.

CDL's prediction on the fifth instance is correct so concepts are not changed. When the sixth instance comes along, CDL predicts it is negative because $\overline{C}$ matches the instance. Upon noticing the prediction is wrong, CDL finds the differences between (1 1 0 1) and previous negative examples {(0 0 0 1) (1 0 0 1) (1 0 0 0) (0 0 1 0)} to be $(\overline{b})$. Thus, the $\overline{C}$ is revised to be $\overline{b}\,\overline{c} \vee \overline{b}\,\overline{a}\,\overline{d}$ and C be $(b \vee c)(b \vee a \vee d)$. Using these new concepts, CDL predicts that the seventh instance is positive (C matches it) but the prediction is wrong. Comparing (1 0 1 0) with the previous positive examples {(1 1 0 1) (1 0 1 1)}, CDL finds the differences to be (d). So the concept C is revised to be $(b \vee c)(d)$, and $\overline{C}$ is revised as $\overline{b}\,\overline{c} \vee \overline{d}$. At this point, the learned concept classifies all the instances correctly.

From this example we can see that CDL does both generalization and discrimination, as the way the concept C is developed. Since CDL achieves the same effects as any bi-directional learning algorithm, it is important to point out the advantages of using complementary discrimination. Unlike most bi-directional learning algorithms, CDL does not require additional domain knowledge specified a priori to bias hypothesis selection that may restrict the concepts that can be learned (e.g., the generalization hierarchies used by LEX). In fact, CDL can learn any subset, either conjunctive or disjunctive, of the instance space although some may be preferred over others. CDL manages its search by "jumping" to a new and reasonable hypothesis in the concept space whenever the concept is revised. This can be seen in the way that differences are found: a

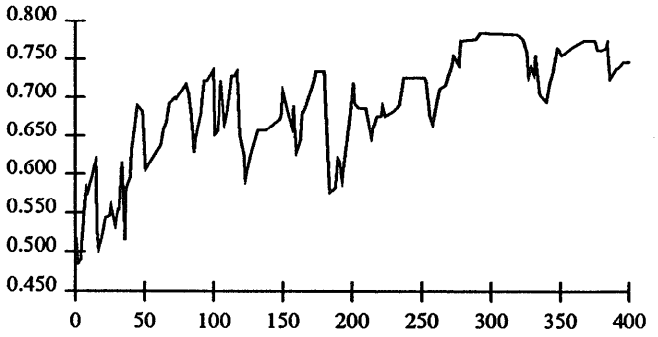


Figure 1: CDL's performance on noisy LED data.

revised concept is neither the most general nor the most specific concept that covers the instances, but a boundary between positive and negative instances that has the simplest syntax structure. The preference for seeking the simplest differences is the backbone of CDL's hypothesis selection.

Finally, although CDL currently implements only learning from mistakes, it can be easily extended to similarity-based learning as well. For example, after correctly predicting the second instance is negative, CDL could, instead of doing nothing, find the similarity between the two negative instances ($\overline{b}d$ in this case) and revise $\overline{C}$ by disjoining $\overline{C}$ to the similarity. CDL can also be extended to select training instances actively; and it has been used in (Shen 1989) to design experiments for autonomous learning from the environment.

Experiments

To illustrate CDL's performance on noisy data, we will report CDL's performance on two typical concept learning tasks: the faulty LED display data and Multiplexor. The former data is noisy; the later data, although noise-free, often causes problems for noise-tolerant algorithms.

Learning from Noisy Data

The LED domain, introduced by Breiman (Breiman *et al.* 1984), is concerned with displaying decimal digits using seven segments, like those on a calculator's display. To introduce noise, consider that each display segment has a 10% chance to be inverted. The task is to recognize the correct digit despite the faulty display. Breiman has shown that for 10% noise rate, the upper bound for the performance of any system is 74%.

We run CDL incrementally on 400 randomly generated instances, uniformly distributed among the 10 digits. In this experiment, each instance has 7 binary-valued attributes representing the seven segments, plus 4 class variables to represent the 10 digits (10 classes). Since CDL is incremental, we evaluate the performance whenever the concepts are revised. The test data are 500 instances that are randomly generated with the same noise rate. CDL's performance curve is shown in Figure 1. The x-axis shows the number of training instances and the y-axis shows the percentage of correct predictions on the 500 test instances. One can see that the prediction rate is generally increasing with the number of instances processed. After 350 instances, CDL's performance is oscillating around 74% (between 72.4% to 77%).[1] Continuing to run CDL on more instances shows that the performance rate will stay in that range. As a point of comparison, the performance of an non-incremental algorithm IWN is 73.3% (or 71.1%) on 400 instances (Tan and Eshelman 1988); and the performance of ID3 is 71.1% using 2000 training instances (Quinlan 1986).

Learning from Data with Exceptional Cases

A Multiplexor is shown in Figure 2. The inputs a and b indicate which input among c, d, e, or f will be output at g. Since variables are binary-valued, there are 64 possible instances in this experiment. The task is to predict the output value given an input vector. It is difficult for noise-free learning algorithms because the relevance of each of the four data-bit attributes is a function of the values of the two address-bit attributes (Utgoff 1988). It is also difficult for noise-tolerant learning algorithms because every training case is equally important (Wilson 1987).

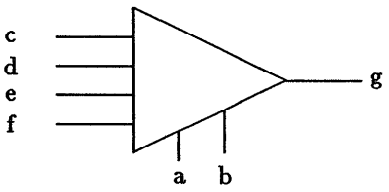
The 64 instances are presented incrementally to CDL as a circulated list. CDL's performance on this task is reported in Figure 2, along with the performances of the algorithms reported in (Utgoff 1988) and IWN. The algorithms with hats are versions in which decision trees are updated only when the existing trees would misclassify the training instance just presented. One can see that CDL's performance is better than others in terms of the number of training events (instances) that are required to reach the 100% prediction rate. Note that time comparison between CDL and others may not be meaningful because CDL is tested on a different machine.

Analysis

Complexity of CDL

In this section, we analyze CDL's complexity based on two assumptions: (1) the training data is noise-free so that the parameter M is set to 100; (2) there are no contradictions when a concept is conjoined with the differences. The second assumption may seem too strong, but we need it at the moment to prove the following theorem.

[1] We think the reason that CDL's performance is sometimes higher than 74% is that the testing data represent only a small subset of the sample space.



Algorithm	Events	Time	Proportion
ID3	53	630.4	100
$\widehat{\text{ID3}}$	61	92.3	100
ID4	384	327.7	63 (not stable)
$\widehat{\text{ID4}}$	384	258.1	50 (not stable)
ID5	57	184.3	100
$\widehat{\text{ID5}}$	74	83.5	100
IWN	320	*	97.6/85.1
CDL	52	10.8	100

Figure 2: CDL's performance in Multiplexor domain.

Theorem 1 *Let i be a new instance wrongly covered by a concept X, and D be the differences found by* DIFFERENCES *with M=100. If X covers* POS *and excludes* NEG, *then the revised concept $X \wedge D$ will cover* POS *and exclude* NEG *and i.*

To see the theorem is true, note that when M=100 DIFFERENCES guarantees that D excludes i and covers POS. $X \wedge D$ will cover POS because both X and D covers POS. $X \wedge D$ will exclude NEG and i because X excludes NEG and D excludes i.

Let I be the number of instances and A be the number of attributes of each instances. The procedure DIFFERENCES will take $O(A \cdot I \log I)$ because DIFF will take $O(A \cdot I)$ to construct, and sorting DIFF and determining the minimum subset of it will take $O(A \cdot I \log I)$. The procedure REVISE will take $O(AYI+Y^2)$ because determining the logical subsumption of each y in Y will take Y^2, and to resolve contradictions takes $A \cdot Y \cdot$OPPOSITE.

To learn a concept that classifies the I instances correctly, the body of CDL will loop no more than I times because of the theorem and the assumption of no contradictions. Thus, the total number of attributes examined by CDL is:

$$\sum_{i=1}^{I} O(A \cdot i \cdot \log i + AYi + Y^2) = O(A \cdot I^2 \cdot \log I + AYI^2 + Y^2 I)$$

If we relax the assumption of no contradictions, then the theorem will not hold because contradictions cause some of the literals in $X \wedge D$ to be modified or deleted. In that case, there may exist instances that are wrongly classified even if all the instances are processed. CDL must go through these "left over" instances and process them again. In all the noise-free experiments we ran, these instances are rare and can be cleaned up quickly.

For all the experiments in this paper, the parameter M in procedure DIFFERENCES has been set to 90. However, to learn from natural data CDL must determine the value of M automatically. One way to achieve this is to initiate M=100 and lower its value when evidence shows that the data is noisy. This can be done at the time of searching for the minimum subset of DIFF. If the data is indeed noisy, then the size of the minimum subset that covers INSTS will grow larger and the some of its elements may only cover a single instance. When this happens, CDL will lower M's value so that the differences that cover single instances will be discarded.

Note that CDL remembers all the previous instances. This could be a weak point if the number of training instances is large. One solution is to limit the memory to recent instances only. We have done some experiments on that matter; however, no formal analysis can be given at this point.

The Quality of Learned Concepts

Besides time complexity, another measure of concept learning algorithms is the quality of the learned concepts. We will consider two aspects here: the predictive power of the concept and the simplicity of the concept.

The concepts learned by CDL have simple structures yet strong predictive power. For example, the concept learned by CDL in the Multiplexor domain is the following:

$$\bar{a}\bar{b}c \vee \bar{a}bd \vee a\bar{b}e \vee abf \vee aef \vee bdf \vee \bar{a}cd \vee \bar{b}ce \vee cdef$$

It can be used to predict the output feature even if the input features are not complete (Fisher 1989). For instance, knowing only that features a, e and f are true, one can conclude g even if the value of b, c and d are unknown. The concept can also be used to predict other features when certain features are known. For instance, knowing that g is true and a is false, one can conclude that c or d must be true because the concept has three literals $\bar{a}\bar{b}c$, $\bar{a}bd$ and $\bar{a}cd$ that contain $\bar{a}$, and to make g true, one of $\bar{b}c$, bd and cd must be true. This seems to be an advantage over ID3 because this predictive ability cannot be easily obtained if the concept is represented by a decision tree.

The concepts learned by CDL also have a close relationship to classification rules. For example, using the variables $c_1c_2c_3c_4$ to represent ten digits, CDL can learn the following concept from an error-free LED display:[2]

[2]The variables U, U_l, U_r, M, B_l, B_r, and B represent the display of *Upper, Up-Left, Up-Right, Middle, Bottom-Left, Bottom-Right* and *Bottom* segment respectively.

$(U_l U_r \overline{M} B_l B_r B \overline{c}_1 \overline{c}_2 \overline{c}_3 \overline{c}_4) \vee (U U_l U_r \overline{M} B_l B \overline{c}_1 \overline{c}_2 \overline{c}_3 \overline{c}_4)$
$\vee (\overline{U M} B_l B_r \overline{B} \overline{c}_1 \overline{c}_2 \overline{c}_3 c_4)$
$\vee (\overline{U}_l U_r M \overline{B}_r B \overline{c}_1 \overline{c}_2 \overline{c}_4 c_3)$
$\vee (\overline{U}_l U_r \overline{B}_l B \overline{c}_1 \overline{c}_2 c_4 c_3) \vee (U \overline{U}_l U_r \overline{M} B_l B \overline{c}_1 \overline{c}_2 c_4 c_3)$
$\vee (\overline{U U_l} \overline{B}_l B \overline{c}_1 \overline{c}_3 \overline{c}_4 c_2)$
$\vee (\overline{U}_r \overline{B}_l \overline{c}_1 \overline{c}_3 c_4 c_2)$
$\vee (U_l \overline{U}_r M B_l \overline{c}_1 \overline{c}_4 c_3 c_2) \vee (\overline{U}_r M B_l B_r \overline{c}_1 \overline{c}_4 c_3 c_2)$
$\vee (U \overline{U}_l \overline{M B}_l \overline{c}_1 c_3 c_2 c_4) \vee (U \overline{U}_l \overline{U}_r \overline{M} \overline{c}_1 c_3 c_2 c_4)$
$\vee (U \overline{U}_l \overline{B}_l \overline{B} \overline{c}_1 c_3 c_2 c_4) \vee (U \overline{U}_r \overline{M} B_l \overline{B} \overline{c}_1 c_3 c_2 c_4)$
$\vee (U_l U_r M B_l B \overline{c}_2 \overline{c}_3 \overline{c}_4 c_1) \vee (U U_l U_r M B_l \overline{c}_2 \overline{c}_3 \overline{c}_4 c_1)$
$\vee (U_l U_r \overline{B}_l B \overline{c}_2 c_4 \overline{c}_3 c_1) \vee (U U_l U_r \overline{B}_l \overline{c}_2 c_4 \overline{c}_3 c_1)$

Note that each conjunction (a single line) is like a rule: When the segments of display matches the description, one can conclude the digit that is displaying. For example, seeing that $U_l U_r \overline{M} B_l B_r B$ is displayed, one can conclude it is a 0 (i.e. $\overline{c}_1 \overline{c}_2 \overline{c}_3 \overline{c}_4$) even if the upper segment is not lit. Because the concepts learned by CDL are very similar to rules, the algorithm can be easily integrated with systems that solve problems and learn by experimentation, as we have shown in (Shen 1989, Shen and Simon 1989). In those cases, CDL will become an unsupervised (or internally supervised) learner because its feedback is from the performance of its problem solver.

Having a simplicity criterion is especially important when the data is noisy because concepts could overfit the data. Fortunately, experiments have shown that CDL manages the tradeoff between simplicity and predictability well. For example, after processing 400 noisy instances in the LED domain, the concept learned by CDL is a DNF of 36 disjuncts. Continuing to run on more new instances will not jeopardize its simplicity nor its predictive ability. We have also run CDL repeatedly on a database of 100 noisy LED instances; the number of disjuncts oscillates between 20 and 30 and the learned concept does not overfit the data.

Conclusion

This paper examines semantically and formally the syntactic duality between generalization and discrimination: generalizing a concept based on similarity between instances can be accomplished by discriminating the complement of the concept based on the difference between instances, and vice versa. Experiments of the CDL algorithm on both perfect and noisy data have shown that exploiting this relation can bring together the advantages of both generalization and discrimination, and can result in powerful learning algorithms. Further studies will be applying CDL to large scale learning tasks, such as learning and discovering new concepts in the CYC knowledge base.

Acknowledgment

I thank Mark Derthick, Doug Lenat, Kenneth Murray, and two anonymous reviewers for their useful comments, and members of the CYC project for their generous support.

References

(Breiman *et al.*, 1984) Breiman, L.; Fredman, L.H.; Olshen, R.A.; and Stone, C.J. 1984. *Classification and Regression Trees.* Wadsworth International Group.

(Feigenbaum and Simon, 1984) Feigenbaum, E.A. and Simon, H.A. 1984. EPAM-like models of recognition and learning. *Cognitive Science*, 8.

(Fisher, 1989) Fisher, D.H. 1989. Noise-tolerant conceptual clustering. In *Proceedings of 11th IJCAI.*

(Lenat, 1977) Lenat, D.B. 1977. The ubiquity of discovery. In *Proceedings of 5th IJCAI.*

(Michalski, 1983) Michalski, R.S. 1983. A theory and methodology of inductive learning. *Artificial Intelligence*, 20.

(Mitchell, 1982) Mitchell, T.M. 1982. Generalization as search. *Artificial Intelligence*, 18.

(Quinlan, 1983) Quinlan, R.J. 1983. Learning efficient classification procedures and their application to chess end games. In *Machine Learning.* Morgan Kaufmann.

(Quinlan, 1986) Quinlan, R.J. 1986. Simplifying decision trees. In *Knowledge Acquisition for Knowledge-based Systems Workshop.*

(Shen and Simon, 1989) Shen, W.M. and Simon, H.A. 1989. Rule creation and rule learning through environmental exploration. In *Proceedings of 11th IJCAI.*

(Shen, 1989) Shen, W.M. 1989. *Learning from the Environment Based on Actions and Percepts.* PhD thesis, Carnegie Melloin University.

(Tan and Eshelman, 1988) Tan, M. and Eshelman, L.J. 1988. Using weigthed networks to represent classification knowledge in noisy domains. In *The Proceedings of the 5th International Machine Learning Workshop.*

(Utgoff, 1988) Utgoff, P.E. 1988. ID5: an incremental ID3. In *The Proceedings of the 5th International Machine Learning Workshop.*

(Vere, 1980) Vere, S.A. 1980. Multilevel counterfatuals for generalizations of relational concepts and productions. *Artificial Intelligence*, 14.

(Wilson, 1987) Wilson, S.L. 1987. Classifier systems and the animat problem. *Machine Learning*, 2:199–288, 1987.

(Winston, 1975) Winston, P.H. 1975. Learning structural descriptions from examples. In *The psychology of computer vision.* MacGraw-Hill.