# Incremental Learning of Independent, Overlapping, and Graded Concept Descriptions with an Instance-Based Process Framework

**David W. Aha**

Irvine Computational Intelligence Project
Department of Information and Computer Science
University of California, Irvine, CA 92717

Technical Report 89–10

23 May 1989

# Incremental Learning of Independent, Overlapping, and Graded Concept Descriptions with an Instance-Based Process Framework

David W. Aha
Department of Information & Computer Science
University of California, Irvine
Irvine, CA 92717 U.S.A.
aha@ics.uci.edu (714) 856-8779

This is an extension of (Aha, 1989), which appears in
*Proceedings of the Sixth International Workshop on Machine Learning*

## Abstract

Supervised learning algorithms make several simplifying assumptions concerning the characteristics of the concept descriptions to be learned. For example, concepts are often assumed to be (1) defined with respect to the same set of relevant attributes, (2) disjoint in instance space, and (3) have uniform instance distributions. While these assumptions constrain the learning task, they unfortunately limit an algorithm's applicability. We believe that supervised learning algorithms should learn attribute relevancies independently for each concept, allow instances to be members of any subset of concepts, and represent graded concept descriptions. This paper introduces a process framework for instance-based learning algorithms that exploit only specific instance and performance feedback information to guide their concept learning processes. We also introduce Bloom, a specific instantiation of this framework. Bloom is a supervised, incremental, instance-based learning algorithm that learns relative attribute relevancies independently for each concept, allows instances to be members of any subset of concepts, and represents graded concept memberships. We describe empirical evidence to support our claims that Bloom can learn independent, overlapping, and graded concept descriptions.

Keywords: supervised concept learning, knowledge-poor, instance-based concept descriptions, independent concepts, overlapping concepts, graded concepts

## 1. Motivation

Consider the problems encountered by expert diagnosticians. Given a description containing large numbers of observations concerning the diagnosee, the diagnostician's responsibilities include (1) determining which information in the complete description is relevant, (2) predicting a coherent diagnosis that explains the diagnosee's symptoms, and (3) predicting (accurately) the *degree* to which the symptoms imply the diagnoses. For example, a physician may observe a patient exhibiting numerous symptoms related to multiple ailments with varying severities, each of which needs to be monitored to determine the accuracy of the diagnosis and the utility of the subsequent treatment(s). Similarly, an automobile mechanic must diagnose and repair a vehicle exhibiting several, possibly related problems. This observation implies that concept learning systems should be able to represent and learn concept descriptions with the following properties:

1. *Independent*: Concepts seldomly share the same set of relevant attributes. Even when they do, attribute relevancies usually differ. Relative attribute relevancies should be independently learned for each concept.

2. *Overlapping*: Instances can be members of any number of concepts (possibly zero), thus allowing overlapping concept descriptions to be represented.

3. *Graded*: Membership in natural concepts is often a graded function. Process models should be able to generate a *confidence* of class membership for each instance.

Unfortunately, few supervised learning algorithms fully resolve all three issues. This is in part due to their (successful) concentration on learning elegant, readable abstractions of the data (Quinlan, 1986; Michalski, Mozetic, Hong, & Lavrac, 1986; Cestnik, Kononenko, & Bratko, 1987). This concentration trades off (i.e., limits) the descriptive capabilities of the concept description representations to solve more difficult concept learning problems. Our view is that this deeper issue must be addressed first; concise abstractions of the data can always be derived from an epistemologically adequate approach later.

Category theorists recognize that exemplar-based models are informationally superior to instance-forgetting abstraction models (e.g., those based on rules, decision trees, or connectionist networks) since they can both recall specific instances and create abstractions upon request (Smith & Medin, 1981; Kahneman & Miller, 1986; Barsalou, 1989). Moreover, studies of human classification behavior indicate that we use both instance-specific and abstraction-level information during recognition and classification tasks (Elio & Anderson, 1981). Based on these premises, we believe that **instance-based** representations for concept descriptions can help to resolve the difficult issues listed above. This problem reduces to finding an instance-based processing model that demonstrates robust concept learning behavior.

This paper introduces an instance-based process framework that specifies robust concept learning algorithms. We then describe a sequence of four comprehensive instantiations of this framework which solve successively more difficult concept learning tasks. Section 4 describes how the current algorithm, Bloom, learns independent, overlapping and graded concept descriptions. Sections 5 and 6 summarize related research and Bloom's limitations respectively.

## 2. The Instance-Based Process Framework

Our approach is called *instance-based learning* (IBL) since it involves making predictions derived from only specific instances and their observed classification behavior. The task addressed in this paper is called *learning from examples* (supervised concept learning). The object of this task is, given a set of training examples annotated with concept membership information, to yield a description for each concept mentioned in the training set. Representations based on abstractions such as first order logic, rules, and decision trees have received significantly more attention in the machine learning literature than has this simpler instance-based representation. However, one of the basic problems with abstraction-based concept description representations is that they can require large updating costs to account for prediction errors. While abstraction-based learning algorithms can form concise abstractions of concepts, they have not yet demonstrated sufficiently robust behavior. Fortunately, IBL algorithms are capable of extraordinarily robust behavior due to the information-based superiority of their representation.

In this paper, instances are represented as a set of $n$ attribute-value pairs that define a point in an $n$-dimensional *instance space.* (We restrict attribute values to be either numeric, Boolean, or nominal-valued.) Each instance is annotated with the *set* of concepts in which it is a member. *Concepts* are unions of regions in the instance space.

IBL algorithms input a sequence of *training instances* and yield a set of *concept descriptions* (one for each concept with member instances in the training set). A concept description contains only a set of instances, information concerning their classification records, and information concerning attribute weight settings. An extension of a concept description maps points in the instance space into the corresponding concept space. These algorithms employ the concept description to yield a *classification* for each instance, defined as the instance's estimated degree of concept membership.

### 2.1 Process Framework

IBL algorithms consist of a framework of the following three components.

1. *Similarity Function:* This computes the *similarity* between a training instance $i$ and the instances in a concept description.
2. *Classification Function:* This inputs the similarity function's results and the classification performance records of a concept description's instances. It yields a classification (i.e., a probabilistic estimate of membership) for $i$ in that concept.
3. *Concept Description Updater:* This maintains records on classification performance, decides which instances to include in each concept description, and maintains the algorithm's attribute weight settings. Inputs include $i$, the similarity results, the classification results, and a current concept description. It yields the modified concept description and updates the attribute weight values.

The algorithms in Section 3 are described in terms of these components.

### 2.2 Performance Measures

We will summarize the IBL algorithms in this paper with respect to the following four performance measures.

1. *Generality*: This is the class of concepts which are describable by the representation and learnable by the process algorithm.
2. *Accuracy*: This is the concept descriptions' classification accuracy for mapping instance space to concept space.
3. *Storage Requirements*: This is the size of the concept descriptions, defined as their number of saved instances.
4. *Incorporation Costs*: This is incurred while updating the concept descriptions with a single training instance and includes classification costs.

We will summarize the generalities of the four IBL algorithms described in Section 3 here and describe their behavior in terms of the other three measures in later sections of this paper. Briefly, we proved that the first two algorithms (Proximity and Growth), when given a good sample of instances, can pac-learn (Valiant, 1984) any concept describable as a union of bounded $n$-dimensional discrete regions in instance space (Kibler & Aha, 1988). NTGrowth, the third algorithm, is believed to have the same representational generality. However, Bloom, the fourth algorithm, can represent overlapping concepts by building each concept description independently, an advantage that is highlighted in this paper.

More specifically, NTGrowth and its predecessors will make prediction errors in those areas of the instance space where concepts overlap. Bloom forms a unique concept description for each concept in which instances are viewed as either members or non-members of each concept in turn. Therefore, it is not susceptible to these same errors. Since each concept is described independently, predictions of concept membership for one concept will not affect predictions for other concepts.

## 3. Instantiations of the Framework

We will use variants of the LED display domain to demonstrate the capabilities of our IBL algorithms. The LED display problem's instance space contains 10 concepts (decimal digits) and 7 Boolean-valued attributes. Each attribute corresponds to a LED light, with value 1 meaning that the light is lit for that instance. See (Breiman, Friedman, Olshen & Stone, 1984) for a more thorough discussion of this domain. Unless otherwise specified, the experimental results reported with variants of the LED display domain used 1000 training instances and 100 separate test instances for each trial.

### 3.1 The Simplest IBL Algorithm

The *Proximity* algorithm (Table 1) is almost identical to the nearest neighbor algorithm (Cover & Hart, 1967).[1] Similarity between instances is defined as the inverse of their Euclidean distance in the instance space. Instances are assigned the classification of their most similar (nearest) neighbor. All training instances are included in the concept descriptions.

We were surprised that Proximity could classify thyroid disease diagnoses with high (97%) accuracy (Kibler & Aha, 1987). Proximity can learn the concepts in the 7-attribute LED display problem. However, it is needlessly storage intensive.

### 3.2 Reducing Storage Requirements

Instances should be included in concept descriptions only when they provide additional information. Therefore, the *Growth*[2] algorithm (Table 1) reduces storage requirements by saving only misclassified instances. It is otherwise identical to Proximity. Growth recorded high classification accuracies and low storage requirements in experiments with several natural domains (Kibler & Aha, 1987; Kibler & Aha, 1988). It needed only 10 of 220 partial

[1] Proximity, like all IBL algorithms, also applys a linear normalization function to all attribute values before processing. The normalization scale is automatically updated when required.

[2] "Growth" refers to the fact that the algorithm's concept description grows in size. This name also refers to a progression in the elaboration of our framework. The current algorithm is therefore named "Bloom".

Table 1: A Summary of Algorithms #1 and #2: Proximity and Growth.

| Framework Component | Proximity | Growth |
|---|---|---|
| Similarity Function | Euclidean Distance | Euclidean Distance |
| Classification Function | Nearest Neighbor | Nearest Neighbor |
| Concept Description Updater | Save All Instances | Save Only Misclassified Instances |

Table 2: A Summary of Algorithms #3 and #4: NTGrowth and Bloom.

| Framework Component | NTGrowth | Bloom |
|---|---|---|
| Similarity Function | Euclidean Distance | Attribute-Weighted Euclidean Distance |
| Classification Function | Nearest Acceptable Neighbor | Nearest Acceptable Neighbor |
| Concept Description Updater | Save Only Misclassified Instances<br>Use only Significantly Good Classifiers<br>Discard Significantly Bad Classifiers | Save Only Misclassified Instances<br>Use only Significantly Good Classifiers<br>Discard Significantly Bad Classifiers<br>Update Attribute Weights |

hypothyroid diagnoses to achieve a 97% classification accuracy. Similar results have been reported with algorithms that either truncate rules (Michalski et al, 1986) or prune decision trees (Breiman et al, 1984).

Growth needs only 1 instance per concept to learn the concepts in the LED display domain. However, Growth is highly sensitive to noise. For example, when each attribute value in the LED instances has a 10% probability of being mislabeled, Growth records a classification accuracy of 63%, far below the domain's optimal Bayesian classification accuracy of 74% (Breiman et al, 1984).

### 3.3 Tolerating Noisy Instances

Methods that truncate or prune tolerate noise by removing statistically insignificant abstractions from concept descriptions (Clark & Niblett, 1989; Breiman et al, 1984; Fisher & Schlimmer, 1988). An extension of Growth named NTGrowth (Noise Tolerant Growth) employs an analogous solution for IBL algorithms. NTGrowth (Table 2) maintains classification records for each saved instance. This performance feedback is used to allow only statistically good classifiers to classify subsequent instances (Aha & Kibler, 1989). NTGrowth also discards statistically poor classifiers from concept descriptions.

NTGrowth recorded a satisfactory (72%) classification accuracy on the noisy LED display domain. It also displayed more graceful degradations (than Proximity and Growth) in classification performance with increasing noise levels (Aha & Kibler, 1989). Furthermore, NTGrowth recorded superb accuracies and even outperformed C4 (a descendant of ID3 (Quinlan, 1986)) on a majority of six complex domains.

Although tolerant of noise, NTGrowth cannot tolerate irrelevant attributes. For example, when applied to an extension of the noise free LED problem that adds 17 irrelevant Boolean attributes, NTGrowth's average classification accuracy is only 67%. In contrast, C4 can classify instances perfectly with the same amount of training (1000 instances).

Table 3: The attribute weight updating algorithm, where the inputs are the current training instance $x$, current concept description $D$, and Classguess($x$) (the predicted classification of $x$). Variable $\lambda$ is the higher relative observed class frequency of $x$'s actual and its predicted class. Variable $y$ is $x$'s most similar neighbor that is also in $x$'s predicted class. Since the instances are normalized, step 3 yields a value in $[0, 1]$.

1. **LET** $\lambda$ = max(ObservedRelativeFrequency(Class($x$)),ObservedRelativeFrequency(Classguess($x$)))
2. **LET** $y = \{d \in D | \forall d' \in D$ {Class($d'$)=Classguess($x$) & Similarity($x$,$d$) $\geq$ Similarity($x$,$d'$)}}
3. **LET** difference = $|x_a - y_a|$
4. **IF** ($x$'s classification was correctly predicted)
   **THEN** increment = (1-$\lambda$)$\times$(1-difference)
   **ELSE** increment = (1-$\lambda$)$\times$difference
5. total-attribute-weight$_a$ = total-attribute-weight$_a$ + increment
6. total-possible-attribute-weight$_a$ = total-possible-attribute-weight$_a$ + (1-$\lambda$)

### 3.4 Tolerating Irrelevant Attributes

Bloom (Table 2) is fourth in a comprehensive progression of instance-based learning algorithms. More specifically, it is an extension of NTGrowth that tolerates irrelevant attributes by learning attribute relevancies independently for each concept. Bloom (1) maintains a separate description for each concept, (2) uses an attribute weighted similarity function, and (3) is otherwise identical to NTGrowth. Bloom classifies the current instance with respect to each concept's description. Whenever the instance is misclassified, it is indexed into the corresponding concept's description.

**Similarity Function**: Bloom defines the similarity of instances $x$ and $y$ with respect to concept $c$ as:[3]

$$\text{Similarity}(c, x, y) = -\sqrt{\sum_{a=1,n} \text{Weight}_{c_a}^2 \times |x_a - y_a|^2} \quad (1)$$

Similarity is concept-dependent. For example, we would expect that, for any tiger $t$ and cat $c$, Similarity($animal, t, c$) is greater than Similarity($pet, t, c$).

**Classification Function**: Bloom classifies each instance $x$ with respect to each concept's description, adding $x$ to the description if it is misclassified. Probability of membership (of $x$ in concept $c$) is defined as:

$$\text{Probability of Membership}(x, c) = \frac{\text{Similarity}(c, x, neg)}{\text{Similarity}(c, x, neg) + \text{Similarity}(c, x, pos)} \quad (2)$$

where $pos$ is a nearest acceptable neighbor of $x$ that is a member of $c$ and $neg$ is a nearest acceptable neighbor of $x$ that is a non-member of $c$.[4]

**Concept Description Updater**: Bloom's updating function is an extension of NTGrowth's (Aha & Kibler, 1989) that learns each concept's $n$ attribute weights through a performance feedback process. Attribute weights are increased when they correctly predict classifications

[3] Bloom applies a (dynamically updated) linear normalization function to all attribute values before processing. All attribute values are scaled to the range $[0, 1]$.

[4] Recall that instances are either members or nonmembers of a given concept. Therefore each concept description groups all non-member instances together, independent of their other concept memberships.

and are otherwise decreased. Weights are derived as follows: (for each attribute $a$)[5]

$$\text{Weight}_{c_a} = max(\frac{\text{total-attribute-weight}_{c_a}}{\text{total-possible-attribute-weight}_{c_a}} - 0.5, 0) \quad (3)$$

The attribute weights are updated after each training instance $x$ is classified. Its most similar neighbor $y$ in the concept description is used to update the weights, as described in Table 3. The total-attribute-weight is incremented by a fraction of that added to the total-possible-attribute-weight. The total-attribute-weight's reward is high when it assists making a correct classification decision and is low otherwise. More specifically, its increment is high when either (1) a correct classification occurs and the instances' attribute values are similar to each other or (2) an incorrect classification occurs and they are dissimilar. Otherwise, the total-attribute-weight's addend is small since the attribute's value did not assist in predicting the correct classification. This algorithm attends to classes with low observed relative frequency in order to overcome highly skewed concept frequency distributions.

The weight-learning algorithm is best explained with an example. For this purpose we will study Bloom's behavior while learning the concept "Ph.D. student" from instances (people) described with three Boolean attributes ("is enrolled","has M.S. degree", and "is married"). Suppose that Bloom has been trained on 4 instances, only one of which was a Ph.D. student (with attribute values <True,True,True>), the resultant total-attribute-weights settings are (0.65,0.65,0.65), and the total-possible-attribute-weights are all 0.75. If the fifth instance is incorrectly classified as a Ph.D. student and has attribute values <False,False,True> (i.e., not enrolled, no M.S., married), then the new total-attribute-weight settings are (0.8,0.8,0.65) and the total-possible-attribute-weights are all 1.0. Finally, if the sixth instance is correctly classified as a Ph.D. student and has attribute values <True,False,False> (i.e., enrolled, no M.S., unmarried), then the new total-attribute-weight settings are (1.6,0.8,0.65) and the new total-possible-attribute-weights are 1.8. This indicates that Bloom has learned that the attribute "is enrolled" is more predictive of the Ph.D. class than either "has M.S. degree" or "is married." Predictive attributes will have higher attribute weights than less relevant attributes.

Updating the attribute weights after each classification continuously changes a concept's similarity function. Thus, Bloom learns similarity functions for each concept independently. Bloom classifies instances from the noise free, 24-attribute LED display domain perfectly.

## 4. Learning Independent, Overlapping, and Graded Concept Descriptions

This section summarizes empirical evidence concerning Bloom's robust learning behavior. We first show that Bloom can learn concepts that have conflicting sets of relative attribute weight settings. Next, we show that Bloom can learn descriptions for concepts that overlap in the instance space. Finally, we show that Bloom can learn concepts whose members have varying probabilities of membership.

[5] Attribute weights are defined in $[0, 0.5]$ rather than $[0, 1]$ because (1) an irrelevant attribute's total weight is expected to be half of its total possible attribute weight and (2) we wanted each irrelevant attribute to have a zero attribute weight.

Table 4: Average percent classification accuracies and storage requirements (25 trials) on four variants of the 24-attribute LED domain. Bloom performs well even when concepts don't share relevant attributes.

| Algorithm | Same Relevant Attributes | | | | Randomly Assigned Relevant Attributes | | | |
|---|---|---|---|---|---|---|---|---|
| | No Noise | | 10 Percent Noise | | No Noise | | 5 Percent Noise | |
| | Accuracy | Storage | Accuracy | Storage | Accuracy | Storage | Accuracy | Storage |
| Proximity | 74.9% | 1000 | 47.2% | 1000 | 78.8% | 1000 | 62.4% | 1000 |
| Growth | 67.7% | 371.6 | 42.0% | 605.5 | 71.6% | 316.3 | 58.0% | 447.9 |
| NTGrowth | 66.7% | 305.7 | 45.8% | 257.8 | 74.5% | 275.0 | 62.4% | 276.7 |
| **Bloom** | **100%** | **239.7** | **68.9%** | **216.0** | **94.2%** | **255.9** | **80.1%** | **247.5** |
| C4 | 100% | | 72.1% | | 87.7% | | 69.6% | |

## 4.1 Learning Independent Concept Descriptions

Bloom learns both a unique set of attribute weights and a unique concept description for each concept. Therefore, it can classify concepts correctly when they are defined independently of each other (i.e., with respect to different sets of relevant attributes).

In order to compare Bloom with the other algorithms, it must yield a "best" classification guess for a given instance, defined as that concept whose probability of membership is highest (i.e., the concept $c$ which yields the maximum value for Equation 2 in Section 3.4).

We applied the four IBL algorithms to two pairs of the 24-attribute LED display problem. The first pair's concepts share the same set of relevant attributes while the latter pair's relevant attributes are randomly selected for each concept. The presence of noise distinguishes variants within each pair. Table 4 summarizes the classification results. As expected, Bloom outperformed the other algorithms in all four applications, both in terms of higher classification accuracies and lower storage requirements.

We also tested C4, a descendant of ID3 (Quinlan, 1986), on these domains. In summary, Bloom's classification accuracies exceeded C4's when the concepts were described by different attributes. This occurred because the decision tree algorithm partitioned these instances into leaves with small numbers of instances, which were subsequently pruned. A learning curve analysis for the fourth domain, where the relevant attributes were randomly selected for each concept and all attribute values were negated with probability 5%, shows that C4 learns more slowly than Bloom, but will probably achieve the same accuracies once these leaves become large enough to be statistically significant. Figure 1 shows the average learning curve through training set sizes of 2000 instances.

## 4.2 Learning Overlapping Concept Descriptions

Unlike our previous algorithms, Bloom allows each instance to be associated with a (possibly empty) **set** of concepts. Instances are interpretated to be members of each of their associated concepts and nonmembers of all others. We extended the LED problem with three additional concepts: (1) even number, (2) prime number, and (3) greater than 6. Table 5 describes the classification accuracies of each concept description when Bloom attempts to learn all 13 concepts simultaneously. The domain again contains 24 attributes (17 irrelevant), where the relevant attributes are selected randomly for each concept, each attribute

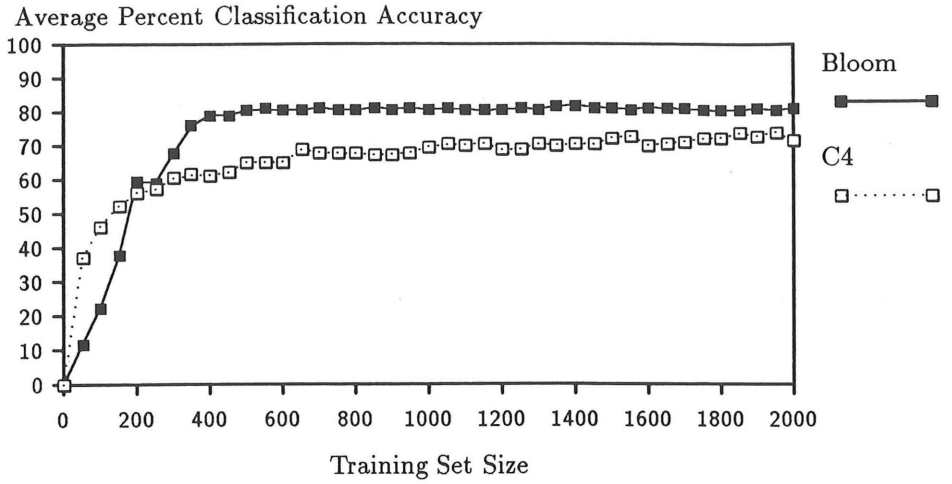


Figure 1: Average learning curves (25 trials) for Bloom and C4 while training on the 24 attribute LED domain when the concepts had randomly assigned relevant attributes and each attribute value had a 5% possibility of being noisy. Bloom learned more quickly than C4, not more accurately, when the relevant attributes were randomly selected for each concept.

Table 5: Average percent classification accuracy results (25 trials) for 13 overlapping concepts. The LED domain contains 24 attributes. The relevant attributes were randomly selected for each of the ten original concepts. Attribute values had a 5% probability of being noisy and instances had a 10% chance of being nonmembers of all concepts.

| Concepts: | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | Even | Prime | $> 6$ | All |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Positives | 85.6 | 92.9 | 89.7 | 91.8 | 87.5 | 90.9 | 89.0 | 92.0 | 89.1 | 89.1 | 71.1 | 73.9 | 71.5 | 85.7 |
| Negatives | 90.7 | 94.6 | 92.5 | 91.3 | 91.7 | 91.6 | 88.9 | 92.6 | 86.3 | 90.7 | 72.3 | 76.6 | 72.0 | 87.1 |
| Average | 90.4 | 94.6 | 92.2 | 91.4 | 91.5 | 91.6 | 88.9 | 92.6 | 86.6 | 90.6 | 71.5 | 76.0 | 72.0 | 86.4 |

has a 5% probability of being mislabeled, and each instance had a 10% probability of being a nonmember of all concepts. The results are described with respect to the accuracy of the individual concept descriptions rather than a "best concept" guess, which makes no sense in this application.

While the average concept description's classification accuracy is high (86.4%), we would like to compare Bloom with other algorithms, such as those that learn decision trees. However, since decision trees partition the instance space into disjoint regions, a correct comparison would require modifying them to accommodate overlapping concepts. Decision tree algorithms were not designed for this task, which involves polythetic prediction (Fisher, 1987). For example, suppose that instances could be members of multiple concepts and that instances did not contain concept membership information for all known concepts. Suppose also that five members of class $A$ and ten instances of class $B$ are classified to node $n$ during training. A test instance $t$ that was classified to node $n$ would subsequently be predicted to be a member of class $B$, or at least have higher probability of being a member of class $B$ than class $A$. However, suppose that all of $n$'s members of class $A$ are also known to be non-members of class $B$ and that none of the class $B$ members contain concept membership information for class $A$. Then $t$ should be predicted to be a member of class $A$ with higher probability than for class $B$.

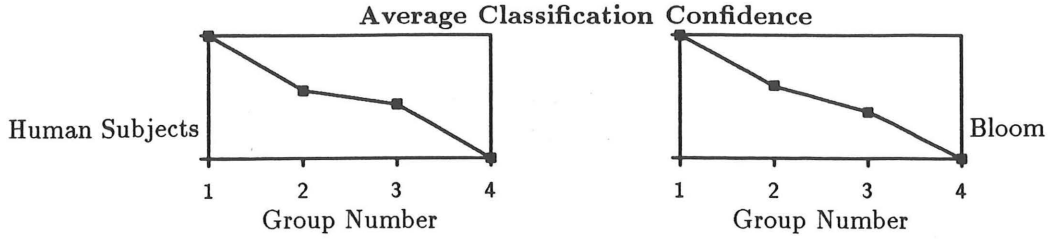


Figure 2: Learning graded concepts: A comparison of the classification confidence results described by Hayes-Roth & Hayes Roth (1977) with Bloom's average behavior (250 trials).

A simple extension of the decision tree approach to solve this problem is to generate a separate decision tree for each overlapping pair of concepts. A better approach may be to instead (1) consider the given set of instances as several sets of instances grouped into positive/negative members of a single concept, (2) modify the node splitting algorithm to choose a split that maximizes information gain across all sets of instances (one set per concept), and (3) make predictions of concept membership separately with respect to that concept's instances. A best classification guess, if required, is simply the class with the highest probability of membership. This notion of generating a separate concept description for each concept is a central theme of our paper.

### 4.3 Learning Graded Concept Descriptions

Psychological studies indicate that categories have *graded* structure. For example, Hayes-Roth and Hayes-Roth (1977) demonstrated that (1) people classify previously unseen prototypes more confidently and accurately than they do previously observed instances, (2) classification performance decreases with an instance's distance from its class' prototype, and (3) higher frequency of presentation during training increases an instance's classification performance.

Bloom was applied to the same data set used in the human subject experiments described by Hayes-Roth and Hayes-Roth (1977), which contains 2 graded concepts (each with one prototype). One dependent measure was classification confidence. In Bloom, the class membership confidence for an instance $x$ in a concept $c$ is defined as follows:

$$\text{Classification Confidence}(x, c) = \frac{\text{Similarity}(c, x, neg)^2 + 1}{\text{Similarity}(c, x, pos)^2 + 1}$$

where (again) *pos* is a nearest acceptable neighbor of $x$ that is a member of $c$ and *neg* is a nearest acceptable neighbor of $x$ that is also a nonmember of $c$.

With this definition, we were able to compare the classification confidences of humans with those of Bloom. Figure 1 displays the results in two graphs, normalized to have the same confidence ranges. This data is broken into 4 groups of test instances: (1) prototypes, (2) near-prototype instances with high learning presentation frequencies, (3) near-prototype instances with low learning presentation frequencies, and (4) far-prototype instances. Both sets of results are averaged over all instances in each group. Bloom's average results (over 250 trials) are highly similar to those recorded with human subjects. Moreover, group classification accuracies (98.6%, 97.8%, 85.2%, and 57.1% respectively) decreased with distance

Table 6: Average percent classification accuracy results on two concepts concerning heart disease.

| Concepts: | Exercised-Induced Angina | | Heart Disease | |
|---|---|---|---|---|
| | Bloom | Frequency | Bloom | Frequency |
| Positives | 71.2% | 32.7% | 72.0% | 45.9% |
| Negatives | 72.9% | 67.3% | 79.0% | 54.1% |
| Average | 72.7% | | 76.0% | |

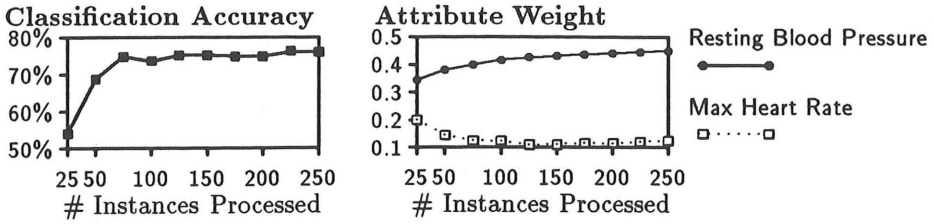


Figure 3: Average learning curve and 2 attribute-weight curves for prediction of the presence of heart disease.

from the prototypes. Finally, classification accuracies and confidences were higher for group 2 instances than for group 3 instances, due to the former's more frequent presentation during learning.[6]

In summary, Bloom closely replicated the behaviors recorded on human subjects by Hayes-Roth and Hayes-Roth (1977). Moreover, the concepts were graded, indicating that Bloom can learn graded concepts.

### 4.4 A Demonstration on a More Complex Domain

Bloom's similarity function is not limited to working with Boolean-valued attributes. The dissimilarity between two values of a numeric-valued attribute is defined as the magnitude of their (normalized) difference. For nominal-valued attributes, the dissimilarity is 0 if the values match or 1 otherwise. We applied Bloom to a database containing heart disease diagnoses (Detrano, 1988) described by 14 attributes (half are numeric and half nominal-valued). Bloom was directed to learn two overlapping concepts simultaneously: predicting (1) whether the patient suffered from exercise-induced angina pectoris and (2) whether the patient had evidence of heart disease. Each trial trained with 250 randomly-selected instances and tested with the remaining 53 instances of the Cleveland data base (Detrano, 1988). The average results (25 trials) are shown in Table 6. As a point of comparison, C4 recorded a 75.4% classification accuracy for predicting heart disease (Kibler & Aha, 1988). Figure 3 displays Bloom's learning curve and two weight-learning curves. Bloom decided that resting blood pressure was a good predictor of heart disease and maximum heart rate is not.

## 5. Related Work

The instance-based process model described in this paper was inspired by research in

[6] Anderson & Kline (1979) indicated that group 2 instances were far from the prototypes when, in fact, they were actually near. While they stated that learning presentation frequency can overcome distance to prototypes, in fact the data indicates that frequency simply mildly assists in classification confidence, but may have a more dramatic effect on classification accuracy.

pattern recognition, categorization theory, and machine learning. IBL algorithms are similar to edited, nearest neighbor pattern classifiers (Cover & Hart, 1967; Hart, 1968; Tomek, 1976). However, they form concept descriptions incrementally, tolerate both noise and irrelevant attributes, and attempt to avoid over-training effects.

Several IBL models have demonstrated psychologically plausible behaviors (Medin & Schaffer, 1978; Hintzman, 1986). In fact, there is compelling evidence suggesting that people use *both* specific instances and abstractions during classification (Elio & Anderson, 1981; Smith & Medin, 1981; Barsalou, 1989). This conclusion led Elio and Anderson to modify ACT to save specific instances and support the partial matching of abstractions. Similarly, Utgoff (1988) saved instances with decision tree leaves to reduce incorporation costs. Volper and Hampson (1987) introduced *specific instance detectors* to improve the convergence rate of perceptron learning in a connectionist framework. These improvements indicate the utility of specific instance information in reducing learning costs in computational learning models.

However, our IBL algorithms do not (yet) form abstractions. Salzberg's (1988) EACH, which inspired the design of Bloom's similarity updating algorithm, is a closely related approach that both saves specific instances and builds abstractions (constrained to be hyper-rectangular in shape). However, EACH employs an ad-hoc parameter in the similarity updating algorithm, uses a graded scale of exemplar strength rather than a significance test to determine which exemplars are used for classification, and does not discard exemplars with poor classification records. Furthermore, Bloom derives inferences solely from instance-specific information and maintains separate descriptions for each concept. Nonetheless, both Bloom and EACH use performance feedback information to update concept descriptions and attribute weights.

Our view of concepts is similar to that of AUTOCLASS II (Cheeseman, Kelly, Self, Stutz, Taylor, & Freeman, 1988) a non-incremental Bayesian clustering algorithm that learns independent, overlapping, and graded clusters whose instances are assumed to be normally distributed. Bloom performs supervised, incremental concept learning, where concept instances need not be normally distributed. However, both approaches represent concepts independently of each other and can learn overlapping, graded concepts.

## 6. Limitations and Future Work

Bloom has several flaws. First, concepts should not be distinguished from other attributes. Blurring them allows the algorithm to express concept relations more flexibly. Second, instances should be associable with any subset of their concepts. It is too much to expect that each instance will be annotated (at presentation time) with membership information (i.e., "positive" or "negative") for all known concepts. Third, concept descriptions should subsequently be updated only when concept information is present in the instance. Finally, we have incorrectly defined similarity such that the similarity between two instances cannot increase with additional attribute comparisons, even when the additional attribute values indicate that these instances are similar. We plan to experiment with variants of Tversky's (1977) contrast model, which defines similarity both in terms of attribute value commonalities *and* (directed) differences, in an attempt to solve this problem.

Breiman (et al, 1984) listed four unsolved problems faced by nearest neighbor, instance-

based concept learning algorithms: (1) sensitivity to the choice of the similarity function, (2) intolerant of nominal-valued attributes, (3) high storage requirements, and (4) inability to derive abstractions. Bloom presents a partial solution to these problems. It learns the similarity function (within the space of weighted Euclidean-based functions), works with nominal-valued attributes, and significantly reduces storage requirements. However, it does not generate abstractions of the data. We plan to focus on methods for generating abstractions in our future research.

Bloom's classification costs involve computing the similarity of new instance $i$ with each instance indexed by each concept description. This involves $O(|C| \times |I| \times |A|)$ attribute inspections, where $C$ is the set of concepts, $I$ the set of incorporated training instances, and $A$ the set of attributes describing the instance space. Average estimates are significantly lower since the number of instances saved in concept descriptions is much lower than $I$. Incorporation costs also involve updating counts on instances and attribute weights. We plan to lower incorporation and classification costs by implementing an indexing scheme that computes similarities only for similar instances.

## 7. Summary

We introduced a process framework for supervised instance-based learning algorithms. We also introduced Bloom, an instantiation of the framework. Bloom is a general algorithm that learns descriptions for independent, overlapping, and graded concepts by attending to each concept's description and attribute relevancies separately. Moreover, Bloom is a relatively robust incremental algorithm that is tolerant of both irrelevant attributes and noise.

Bloom relaxes the constraints for supervised learning tasks. For example, instances are allowed to be members of any subset of concepts (perhaps none). However, it still maintains several unwarranted assumptions which are not required by the framework. For example, instances should be describable by any subset of attributes, the similarity function should obviate the need for special measures to tolerate missing attribute values, and concepts should not be distinguished from attributes. Nonetheless, Bloom advances the capabilities of instance-based algorithms. It can incrementally learn independent, overlapping, and graded concept descriptions. We plan to continue scaling up the instance-based approach to solve more difficult concept-learning tasks in the future.

## Acknowledgements

Many thanks to Marc Albert, Dennis Kibler, Dale McNulty, John Gennari, David Ruby, and John Allen for reviewing earlier drafts of this paper. We would also like to thank Dr. Robert Detrano, M.D., Ph.D., for donating the Cleveland Clinic Foundation's database containing heart disease diagnoses. This database is one of 40 available from the UCI Repository of Machine Learning Databases.

## References

Aha, D. W. (1989). Incremental, Instance-Based Learning of independent and graded concept descriptions. To appear in *Proceedings of the Sixth International Workshop on Machine Learning.* Ithaca, NY: Morgan Kaufmann.

Aha, D. W., & Kibler, D. (1989). Noise-tolerant instance-based learning algorithms. To appear in *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence.* Detroit, MI: Morgan Kaufmann.

Anderson, J. R., & Kline, P. J. (1979). A learning system and its psychological implications. In *Proceedings of the Sixth International Joint Conference on Artificial Intelligence* (pp. 16–21). Tokyo, Japan: Morgan Kaufmann.

Barsalou, L. W. (1989). On the indistinguishability of exemplar memory and abstraction in category representation. To appear in T. K. Srull & R. S. Wyer (Eds.), *Advances in Social Cognition.* Hillsdale, NJ: Erlbaum.

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and regression trees.* Belmont, CA: Wadsworth International Group.

Cestnik, B., Kononenko, I., & Bratko, I. (1987). ASSISTANT-86: A knowledge-elicitation tool for sophisticated users. In I. Bratko & N. Lavrac (Eds.), *Progress in machine learning.* Bled, Yugoslavia: Sigma Press.

Cheeseman, P., Kelly, J., Self, M., Stutz, J., Taylor, W., & Freeman, D. (1988). AutoClass: A Bayesian classification system. In *Proceedings of the Fifth International Conference on Machine Learning* (pp. 54–64). Ann Arbor, MI: Morgan Kaufmann.

Clark, P. E., & Niblett, T. (1989). The CN2 induction algorithm. *Machine Learning, 3,* 261–284.

Cover, T. M., & Hart, P. E. (1967). Nearest neighbor pattern classification. *Institute of Electrical and Electronics Engineers Transactions on Information Theory, 13,* 21–27.

Detrano, R., M.D. (1988). International application of a new probability algorithm for the diagnosis of coronary artery disease. Unpublished Manuscript.

Elio, R., & Anderson, J. R. (1981). The effects of category generalizations and instance similarity on schema abstraction. *Journal of Experimental Psychology: Human Learning and Memory, 7,* 397–419.

Fisher, D. (1989). Knowledge acquisition via incremental conceptual clustering. *Machine Learning, 2,* 139–172.

Fisher, D. H., & Schlimmer, J. C. (1988). Concept simplification and prediction accuracy. In *Proceedings of the Fifth International Conference on Machine Learning* (pp. 22–28). Ann Arbor, MI: Morgan Kaufmann.

Hart, P. E. (1968). The condensed nearest neighbor rule. *Institute of Electrical and Electronics Engineers Transactions on Information Theory, 14,* 515–516.

Hayes-Roth, B., & Hayes-Roth, F. (1977). Concept learning and the recognition and classification of exemplars. *Journal of Verbal Learning and Verbal Behavior, 16,* 321–338.

Hintzman, D. L. (1986). "Schema abstraction" in a multiple-trace memory model. *Psychological Review, 93,* 411–428.

Kahneman, D., & Miller, D. T. (1986). Norm theory: Comparing reality to its alternatives. *Psychological Review, 93,* 136–153.

Kibler, D., & Aha, D. W. (1987). Learning representative exemplars of concepts: An initial case study. In *Proceedings of the Fourth International Workshop on Machine Learning* (pp. 24–30). Irvine, CA: Morgan Kaufmann.

Kibler, D., & Aha, D. W. (1988). Comparing instance-averaging with instance-filtering learning algorithms. In *Proceedings of the Third European Working Session on Learning* (pp. 63–80). Glasgow, Scotland: Pitman.

Medin, D. L., & Schaffer, M. M. (1978). Context theory of classification learning. *Psychological Review, 85,* 207–238.

Michalski, R.S., Mozetic, I., Hong, J., & Lavrač, N. (1986). The multi-purpose incremental learning system AQ15 and its testing application to three medical domains. In *Proceedings of the Fifth National Conference on Artificial Intelligence* (pp. 1041–1045). Philadelphia, PA: Morgan Kaufmann.

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

Salzberg, S. (1988). *Exemplar-based learning: Theory and implementation* (Technical Report TR-10-88). Cambridge, MA: Harvard University, Center for Research in Computing Technology.

Smith, E. E., & Medin, D. L. (1981). *Categories and concepts.* Cambridge, MA: Harvard University Press.

Tomek, I. (1976). An experiment with the edited nearest neighbor rule. *Institute of Electrical and Electronics Engineers Transactions on Systems, Man, and Cybernetics, 6*, 448–452.

Tversky, A. (1977). Features of Similarity. *Psychological Review, 84*, 327–352.

Utgoff, P. E. (1988). ID5: An incremental ID3. In *Proceedings of the Fifth International Conference on Machine Learning* (pp. 107–120). Ann Arbor, MI: Morgan Kaufmann.

Valiant, L. G. (1984). A theory of the learnable. *Communications of the Association for Computing Machinery, 27*, 1134–1142.

Volper, D. J., & Hampson, S. E. (1987). Learning and using specific instances. *Biological Cybernetics, 57*, 57–71.