Instance-based prediction of real-valued attributes

DENNIS KIBLER, DAVID W. AHA, AND MARC K. ALBERT

Department of Information and Computer Science, University of California, Irvine, CA 92717, U.S.A.

Received September 26, 1988

Revision accepted January 27, 1989

Instance-based representations have been applied to numerous classification tasks with some success. Most of these applications involved predicting a symbolic class based on observed attributes. This paper presents an instance-based method for predicting a *numeric* value based on observed attributes. We prove that, given enough instances, if the numeric values are generated by continuous functions with bounded slope, then the predicted values are accurate approximations of the actual values. We demonstrate the utility of this approach by comparing it with a standard approach for value prediction. The instance-based approach requires neither ad hoc parameters nor background knowledge.

Key words: instance-based concept descriptions, prediction, incremental learning algorithms, continuous functions, learning theory.

Des représentations basées sur des instances ont été appliquées avec un certain succès à de nombreuses tâches de classification. La plupart de ces applications nécessitait la prévision d'une classe symbolique basée sur des attributs observés. Cet article présente une méthode basée sur des instances servant à prédire une valeur *numérique* en fonction d'attributs observés. Les auteurs démontrent que, si les valeurs numériques sont produites par des fonctions continues avec pente limitée, les valeurs calculées constituent des approximations précises des valeurs réelles, compte tenu d'un nombre suffisant d'instances. Les auteurs illustrent l'utilité de cette méthode en la comparant à une approche standard de prévision des valeurs. La méthode basée sur les instances n'exige aucun paramètre ad hoc ou connaissance d'arrière-plan.

Mots clés : description de concepts basés sur des instances, prévision, algorithmes d'apprentissage incrémentiel, fonctions continues, théorie de l'apprentissage.

[Traduit par la revue]

Comput. Intell. 5, 51-57 (1989)

1. Introduction

Instance-based learning (IBL) strategies represent concept descriptions as a set of instances, where each instance is described in terms of a set of attribute–value pairs. IBL techniques have been applied to several learning problems, including speech recognition (Bradshaw 1987), word pronunciation (Stanfill and Waltz 1986; Stanfill 1987), handwritten symbol recognition (Kurtzberg 1987), thyroid disease diagnosis (Kibler and Aha 1987), and the cart–pole balancing problem (Connell and Utgoff 1987). In each case, IBL techniques were shown to be computationally inexpensive methods for solving classification tasks. Most IBL applications involved the prediction of symbolic values. However, Connell and Utgoff applied their CART system to predict values for a numeric domain, namely the degree of desirability of instances (which represent states in their domain). In this paper we present a related method for applying instance-based techniques for predicting numeric values. Additionally, we present theoretical and empirical arguments to support our claims concerning the generality of these techniques.

Most of the published work on learning from examples tasks involved the prediction of symbolic values (Vere 1980; Mitchell 1982; Quinlan 1987; Michalski *et al.* 1986). In many cases, numeric attribute domains were transformed to symbolic domains to simplify the learning task. This transformation often required the assistance of an expert. In many domains, however, experts do not exist or are sufficiently scarce to make the transformation task difficult. Some systems, such as STAGGER (Schlimmer 1987), automatically perform this transformation process, but they incur large computational costs and do not guarantee success. IBL techniques do not require these transformations.

Instance-based techniques are good methods to use for symbolic value-prediction tasks because they are extremely simple to apply, they allow for the efficient incremental processing of training instances, and they are highly tolerant of noise in the predictor attributes (Stanfill 1987; Aha and Kibler 1989). They are also good methods for numeric value-prediction tasks. We can prove that instance-based prediction (IBP) techniques are appropriate when given noise-free instances (see Sect. 3) and they are applicable to a large set of value-prediction tasks. In a sense that we will make precise, any continuous function with bounded slope is learnable by these techniques.

The set of continuous functions contains a huge class of naturally occurring functions. Most physical laws are differential equations whose solutions are differentiable functions and, hence, continuous. People estimate the value of continuous and real-valued functions both consciously and unconsciously. Unconsciously, we estimate physical processes during such daily events as balancing, walking, eating, pouring, cooking, driving, throwing, and catching. Our ability at sports rests on the ability to predict, with some accuracy, the positions and velocities of moving bodies as well as the ability to coordinate and execute our own movements. Mispredictions, such as expecting another step while climbing stairs or lifting heavy objects which appear to be light, can yield calamitous results. Consciously, we estimate the value of houses and cars, the temperature of the air and water, the amount of time that tasks take, the distance between locations, and what we can fit into a suitcase. Consequently, the ability to estimate real-valued functions is important in everyday activities.

TABLE 1. The proximity algorithm experiment (C = concept set)

```
∀t ∈ Training set:
  • $C \leftarrow C \cup \{\text{normalize}(t)\}$
∀t ← normalize(t′), t′ ∈ Test set:
  • $\forall c \in C \{c \neq t\}$: calculate Similarity(t,c)
  • Let Sim ⊆ C be the set of N% most similar instances of C to t
  • Let Sum = $\sum_{c \in \text{Sim}} \text{Similarity}(t, c)$
  • Then F-estimate(t) = $\sum_{c \in \text{Sim}} \frac{\text{Similarity}(t, c)}{\text{Sum}} \times F(c)$
```

In Sect. 2 we illustrate our algorithms and compare IBP techniques with linear regression techniques for predicting real-valued functions. The latter is an alternative prediction method that is most applicable to those functions expressible as a linear combination of their attributes. IBP techniques are more generally applicable to the class of locally linear functions. One advantage of linear regression is that it tolerates noise. In Sect. 2 we introduce a simple technique (later applied in our experiments) that allows instance-based approaches to tolerate noise when predicting numeric values. (General techniques for tolerating noise in instance-based symbolic classification tasks are described in Aha and Kibler (1989). Section 3 summarizes our theoretical justifications for applying these techniques to numeric value-prediction tasks. We provide empirical justifications in Sect. 4. We conclude in Sect. 5 with a discussion and evaluation of IBP.

2. Algorithms and illustrations

Instance-based learning and prediction algorithms have predominantly been applied to symbolic classification tasks. In these tasks each instance is generally represented as a set of attribute-value pairs, where the values are either numeric or nominal. The value to be predicted is always nominal. A concept is a subspace of the instance space.

The problem to be solved by symbolic classification algorithms is to determine a function F that, given an instance, yields the concept of the instance. More precisely, let C be a set of concepts, I be an instance space with n attributes, and V_i be the set of values in attribute domain $i(1 \leq i \leq n)$. Then

$$F(\langle x_1, \ldots, x_n\rangle) = C_i$$

where $\forall_i(1 \leq i \leq n) \{x_i \in V_i\}$ and $C_i \in C$.

Given a set of instances, learning algorithms typically generate a summary description or predicate for each symbolic concept. These predicates can be applied to a new instance to yield a classification. In contrast, instance-based approaches represent a concept as a set of instances. A new instance is classified by some form of best-match with existing concepts.

For example, Kibler and Aha (1987) have applied several IBL algorithms to two of Quinlan's thyroid disease databases (Quinlan 1987). The instances were described in terms of 26 attributes. The concepts consisted of the set

{hypothyroid, sick-euthyroid, negative}.

Each learning algorithm was given a *training set* of instances from which it derived a *concept description*, which is also a set of instances. The concept description was subsequently tested on a *test set* of disjoint instances. The nearest neighbor classification algorithm was used for all the learning algorithms. For each test instance t, it guessed that $F(t) = F(n)$, where n is t's most similar neighbor in the concept set (similarity is defined here as the negation of the Euclidean distance between two instances in the n-attribute instance space). We measured each algorithm's performance by recording the percentage of test instances that were classified correctly by the concept description. In effect, the extension of the concept description (derived by applying the similarity and classification to the concept description) described each algorithm's guess of F with respect to the instances on which it was trained.

TABLE 2. Similarity of two normalized instances (n dimensions)

$$\text{Similarity}(t, c) = \sum_{i=1}^{n} \text{Simil}(i(t), i(c))$$

where $\text{Simil}(x, y) = 1.0 - |x - y|$, and
$i(t)$ yields the attribute value of instance t in dimension i

In this paper we are concerned with instance-based methods for predicting numeric rather than symbolic values. More specifically, if we let R be a numeric domain, then we can describe F for predicting numeric domain values as follows:

$$F(\langle x_1, \ldots, x_n\rangle) = r_i$$

where $\forall_i(1 \leq i \leq n)$, $\{x_i \in V_i\}$ and $r_i \in R$.

Connell and Utgoff (1987) recently applied IBP techniques to the cart-pole balancing problem. Their system predicts state desirability, whose values are continuous from -1 (undesirable) to 1 (most desirable). Saved instances have values of either -1 or 1. All other instances' degree of desirability are derived via a weighted-similarity function of the saved instances' desirability values. In our case, saved numeric domain values can have any range and any value within that range.

We chose to use the simplest instance-based learning algorithm, called the *proximity* algorithm, for the experiments in this paper. The proximity algorithm simply saves all training instances in the concept set. The IBP method employed in the experiments is detailed in Table 1. The normalization algorithm maps each attribute value into the continuous range 0–1, assuring that all attributes are assigned equal classification salience. Assuming that each attribute counts equally is not necessarily best, but is, at least, fair. The problem of finding the appropriate weights for attributes is another example of the unsolved credit assignment problem. The estimate $F(t)$ for test instance t is defined in terms of a weighted-similarity function of t's nearest neighbors in the concept set. The similarity of two normalized instances is defined in terms of their Euclidean distance, as detailed in Table 2.

The classification algorithm employed in our experiments is a variant of Shepard's function (Connell and Utgoff 1987) that, instead of using all concept instances to assist in classifying a new instance, uses only the subset of the neighboring concept instances for numeric value prediction. The underlying assumptions of the linear regression model is that F is approximately linear. In contrast, using only a few neighboring instances for classification assumes that the concept function is locally linear, a much weaker assumption. Unfortunately, our techniques become more sensitive to noise as fewer instances are used.

If the function is linear and without noise, then our function produces the same effect as interpolation. For example, if the domain is the real numbers, then the predicted value of a test instance is the weighted average of the F values of its two closest observed instances. Figure 1 illustrates an application of our techniques where F is a polynomial function.[1] Included in Fig. 1 are two approximations of F, one by IBP using 10 training instances and the other by a linear regression model derived from the same training set. Linear regression techniques, of course, are not meant to be applied to nonlinear functions. This simply demonstrates a case where the IBP method more accurately approximates the function than does linear regression.

In the general case where there are n attributes, one might expect to use the $n + 1$ nearest neighbors. Instead we choose to use N% of the nearest values. This allows us to tolerate some noise and yet does not demand that we assume the function is globally linear. We take an "average" hyperplane defined by these instances to predict the value of the unknown.

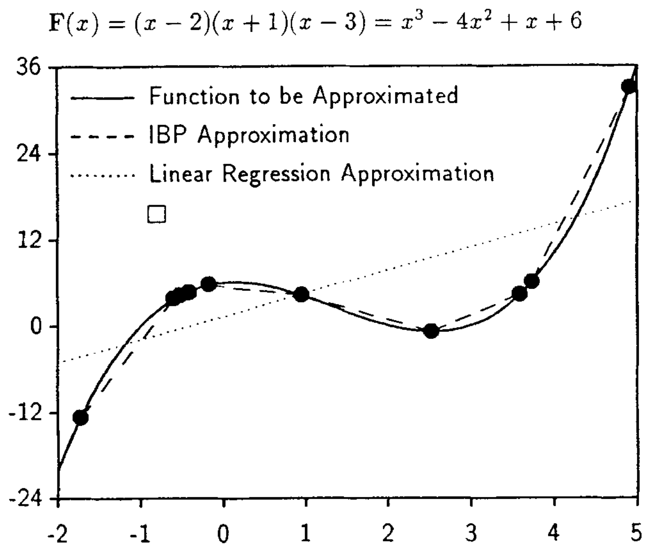

FIG. 1. Approximating a typical nonlinear function. (Both approximations were derived from the 10 instances shown above.)

3. Theoretical justification

The goal of this section is to demonstrate that instance-based techniques can learn to predict the value of any continuous function with bounded derivative. We will define what we mean by "learn" in a way that preserves the spirit of the Valiant (1984) definition of learnability, but generalizes it so that it makes sense in the context of real-valued attribute prediction. To do this we first establish that a sufficiently large random sample of instances chosen in accordance with any fixed probability distribution represents a good sampling of instances. Then we demonstrate that, given a good sample of instances, a piecewise linear function can be generated which will closely approximate the unknown function.

3.1. Coverage lemma

Here we establish that a large-enough sample gives a good coverage of the domain.

The following lemma shows that, given any probability distribution for the points in the unit interval [0,1], only polynomially many examples need to be chosen in order to (with a high level of confidence) "cover" [0,1] well, except possibly on a small (low probability) subset.

When we talk about the "size" of a set, we mean its probability. We will use "length" to refer to the Euclidean length of an interval.

Lemma 1

Let $0 < \gamma, \epsilon, \delta < 1$ and let the space [0,1] have any probability distribution. Then there exists a polynomial p such that if $m > p\left(\frac{1}{\gamma}, \frac{1}{\epsilon}, \frac{1}{\delta}\right)$ points from [0,1] are sampled according to the given distribution, then, with confidence greater than $1 - \delta$, every point of [0,1] will be within γ of some sample point (except on some set of size less than ϵ).

[1]We use a modified definition of IBP approximation for one-dimensional applications. Instead of making predictions from a set of nearest neighbors, the IBP approximation bases approximations on the two "surrounding" sample instances and uses the nearest neighbor approach for instances that are not surrounded. See Theorem 1 for details.

Proof

Let γ, ϵ, and δ be arbitrary positive numbers less than 1. We prove this lemma by partitioning the unit interval into k equal-lengthed subintervals, each of length less than γ. By selecting enough instances we assure that, with high probability, each subinterval of sufficient probability will contain at least one of the selected instances.

Let $k > 1/\gamma$. Let S_1 be the set of subintervals $[x_i, x_{i+1}]$ such that $P([x_i, x_{i+1}]) \geq \epsilon/k$. Let S_2 be the set of the remaining subintervals.

The probability that an arbitrary instance $i \in [0,1]$ selected according to the given probability distribution will not lie in some selected subinterval in S_1 is at most $(1 - \epsilon/k)$. The probability that none of the m sample instances will lie in a selected subinterval in S_1 is at most $(1 - \epsilon/k)^m$. The probability that any subinterval in S_1 is excluded by all m sample instances is at most $k(1 - \epsilon/k)^m$. Since

$$\left(1 - \frac{\epsilon}{k}\right)^m < e^{\frac{-m\epsilon}{k}}$$

then

$$k\left(1 - \frac{\epsilon}{k}\right)^m < k\, e^{\frac{-m\epsilon}{k}}$$

We assure that this probability is small by forcing it to be less than δ. We solve for m as follows:

$$k\, e^{\frac{-m\epsilon}{k}} < \delta$$

$$e^{\frac{-m\epsilon}{k}} < \frac{\delta}{k}$$

$$-\frac{m\epsilon}{k} < \ln\left(\frac{\delta}{k}\right)$$

$$m > -\frac{k}{\epsilon}\ln\left(\frac{\delta}{k}\right)$$

$$m > \frac{k}{\epsilon}\ln\left(\frac{k}{\delta}\right)$$

Consequently, with probability greater than $1 - \delta$, each subinterval in S_1 contains some sample instance. Also, the total size of all the subintervals in S_2 is less than $(\epsilon/k)k = \epsilon$. Since each instance of [0,1] is therefore in some subinterval of S_1, except on a set of size no more than ϵ, then, with probability greater than $1 - \delta$, an arbitrary instance of [0,1], selected according to the given probability distribution, is within γ of some sample instance (except on a set of size no more than ϵ). ∎

The proof of Lemma 1 generalizes to any bounded region in R^n (i.e., it guarantees that by picking enough samples, we can assure that we will probably get a good coverage of any nice region).

3.2. Instance-based prediction theorems

Here we prove that, given a good sample of instances, IBP can generate a piecewise linear function that is a good approximation to an unknown continuous function.

We define the error of a real-valued function $\bar{f}$ in predicting a real-valued function f, at a point x, as $|f(x) - \bar{f}(x)|$.

Let f be a real-valued function. We define B_f to be the least upper bound of the absolute value of the slope between any two points on the curve of f. If B_f is finite then we say that f has *bounded slope*. If C is a class of functions in which each $f \in C$ has bounded slope, then we say that C has bounded slope. As an example, the class of continuously differentiable functions on [0,1] has bounded slope.

Definition

Let C be a class of functions for which each $f \in C$ is a function with bounded slope from the unit hypercube in R^n to R. C is *polynomially learnable* from examples if there exists an algorithm A and a polynomial p such that, for any $0 < \gamma, \epsilon, \delta < 1$ and $B > 0$, given an $f \in C$ with $B_f \leq B$, if more than $p\left(\frac{1}{\gamma}, \frac{1}{\epsilon}, \frac{1}{\delta}, B\right)$ examples are chosen according to any given probability distribution on [0,1], then, with confidence at least $1 - \delta$, A will output an approximation of f with error no more than γ (except on a set of size less than ϵ). Also, A halts in time bounded by a polynomial in the number of examples.

The following theorem demonstrates that continuous, real-valued functions with bounded slope in the unit interval [0,1] are polynomially learnable. Note that the class of functions with bounded slope includes the class of continuously differentiable functions. Extensions to multivalued functions of multiple arguments is standard, requiring only a working knowledge of advanced calculus.

Theorem 1

Let C be the class of continuous, real-valued functions on [0,1] with bounded slope. Then C is polynomially learnable by IBP techniques.

Proof

Let f be a continuous function on [0,1]. Let γ, δ, and ϵ be arbitrary positive numbers less than 1. We will guarantee that f does not vary much on a small interval by using the bound B_f. In particular, we apply Lemma 1 with $\gamma' = \gamma/2B_f$. (We assume, without loss of generality, that $B_f \geq 1$). The lemma guarantees that, if m is large enough, then with confidence greater than $1 - \delta$, every point of [0,1] will be within γ' of one of the selected m points, except on a set of size less than ϵ.

We define the approximation function $\bar{f}(x)$ of $f(x)$ as follows. For each of the m sample points, let $\bar{f}$ be defined as the sample value. On nonsample points, let $\bar{f}$ be defined as the linear interpolation of its surrounding neighbors. If a nonsample point p is not surrounded, then let $\bar{f}(p) = f(x)$, where x is the closest neighbor in the sample.

Let x be an arbitrary point in [0,1] that is contained in some subinterval $s_1 \in S_1$, where S_1 is defined as in Lemma 1. We shall consider first the case where x is surrounded by sample instances x' and x'', where x' is the nearest neighbor of x in s_1. Then

$$f(x) = f(x') + s(x', x) \times |x' - x|$$

and

$$\bar{f}(x) = \bar{f}(x') + \bar{s}(x', x'') \times |x' - x|$$

where $s(x', x)$ is the slope of f between x' and x and $\bar{s}(x', x'')$ is the slope of $\bar{f}$ between x' and x''.

Since B_f is an upper bound on the absolute value of the slope between any two points for $f(x)$, it is also an upper bound on the absolute value of the slope between any two points for $\bar{f}(x)$. Therefore,

$$|f(x) - f(x')| = |s(x', x)| \times |x' - x| < B_f \times \gamma/2B_f = \gamma/2$$

and

$$|\bar{f}(x) - \bar{f}(x')| = |\bar{s}(x', x'')| \times |x' - x| < B_f \times \gamma/2B_f = \gamma/2$$

Since $\bar{f}(x') = f(x')$, we have, by triangular inequality,

$$|f(x) - \bar{f}(x)| < \gamma/2 + 0 + \gamma/2 = \gamma$$

The algorithm clearly runs in time proportional to the number of sample instances since it requires only that we find the two surrounding sample instances for x and compute the equation of the line that connects them.

The second case, where x is not surrounded by sample instances, can be handled similarly. ∎

Notice that the above proof also shows that if we restrict ourselves to the class of continuous functions with slope bounded by a single constant B, then the number of examples required (and consequently the time required) to learn any concept in the class is dependent only on the level of confidence and accuracy desired, and is independent of the particular element of the class to be learned. This permits us to learn concepts in the class, and know when it is safe to stop, without knowing the maximum value of the slope for the particular concept being learned. On the other hand, it requires that we know the value of a bound on the slope for the entire class of concepts that we would like to learn.

Using an extension of Lemma 1 and a proof similar to that used for Theorem 1, we can prove Theorem 2.

Theorem 2

The class of continuous functions on a closed and bounded n-dimensional space with bounded derivative is polynomially learnable by IBP techniques.

Note that any piecewise linear curve and any function with a continuous derivative has a bounded slope.

This result indicates that, given an open domain, one needs to require some constraints on the "wildness" of the function to assure that the time to learn is polynomially

4.1. Expert vs. user mode of running LEW

The oval boxes in Fig. 1 denote activities of the learning process in which a human user is involved. The bold arrows denote flow of control in the expert mode, while the normal arrows denote the user mode.

In the expert mode LEW accepts a cue from the expert/teacher, and performs generalization using its inference engine. The results of generalization are verified by the expert. If the volume of cues produced by LEW's inference engine during generalization is very large, we propose to use a statistical quality control procedure (Constant *et al.* 1989). In this manner, a knowledge base is created by LEW in cooperation with a teacher who submits the examples. The results of learning do not depend on the order in which the cues are given to LEW by the expert. The expert mode, thus, allows explicit modifications to the knowledge base: answering unanswered user questions, asserting new cues, and confirming or disconfirming potentially erroneous generalizations. Disconfirmed generalizations are stored in a list which is consulted during later learning episodes to prevent duplicating old mistakes. Finally, the direct modification of the knowledge base through the assertion or retraction of a cue by the expert invokes the learning algorithm, which may cause a reorganization of the entire knowledge base.

In the user mode, the user asks LEW questions (of course, without giving the answers at the same time), just as if an expert system was used. If LEW succeeds in putting together all the components of the answer deemed necessary by the question-answering algorithm (Constant *et al.* 1987) to answer the question, the planning component (which will not be further described in this section) synthesizes the answer. The user has the option to request a high-level answer or to immediately request the most detailed answers or to step along a hierarchy through which increasing or decreasing levels of detail in the answers can be obtained. The latter feature equips LEW with a rudimentary explanation facility. (This same facility is used as a debugging tool in the expert mode.)

Should it prove impossible to produce an answer, LEW will store the question and pass it on to the expert who updates the knowledge base at regular time intervals. This design goes some way towards meeting the objective of a closed-loop conceptual learner (Michalski and Watanabe 1988), in which the results of learning influence the learning process itself.

4.2. An algorithmic description of LEW

In the sequel, we introduce the basic concepts and operations necessary to describe the algorithm on which the learning in LEW is based. As remarked above, a *cue* consists of a question–answer pair with an optional environmental component, and is represented as follows:

- cue(Cluster_Index, Cue_Index, Environment, Question, Answer).

The Question consists of a marker indicating the question type — such as "How do I" or "Why" — and a phrase representing its propositional content. Both the Answer and the Environment are lists of phrases. In certain cases, the Answer may comprise a list of lower-level questions, the answers to which are taken to jointly constitute the given Answer. The Environment, if nonempty, can specify domain knowledge to constrain generalization or, in planning applications, describe the initial environment that is consulted to see if an action's preconditions are met.

When LEW is supposed to learn plans, it is also provided with a basic repertoire of *actions* which have the following form:

- action(Action_Index, Input_Environment, Action, Output_Environment).

Each action predicate is to be interpreted as asserting that performing Action in the Input_Environment yields a state described by Output_Environment. Input_Environment and Output_Environment are lists of phrases, while Action is a single phrase. (The Action_Index will be omitted in several illustrations below.)

A *similarity cluster* is a set of cues each of which is linked to at least one other cue in the cluster by virtue of having a question or an answer in common with it or of being synonymous with it or of belonging to the *same type* (see below). The generalization procedure is only applied within one such cluster because, intuitively, generalizing on sets of less similar cues would lead to a proliferation of far misses.

Domain-specific *synonyms* are supplied by the expert or inferred on the basis of single differences between cues. Thus, LEW creates a synonym in the following two situations: either two cues have the same question (answer) and the two answers (questions) differ by only one word and the two environments are identical[2] or differ by that same word, or two identical cues have environments that differ by only one word. In either case, LEW assumes such cues themselves to be synonymous and collapses them into one.

Types are either specified by the expert before learning begins or created by LEW as a result of learning. We define as type two pairs of words that realize the difference between two cues in which the two questions differ by exactly one word, and the two corresponding answers differ by only one other word, and the two corresponding environments differ by at most these two words.

For example, in the following cues let the first pair represent the question, and the second pair represent the answer:

$$c_1 = [[a, a], [A, A]]$$
$$c_2 = [[a, b], [A, B]]$$

c_1 and c_2 differ by a type, and $(\langle a, A\rangle, \langle b, B\rangle)$ is the type that realizes the difference.

Naturally, learning proceeds more smoothly and generalizations are more powerful in the presence of a well-developed, hierarchized type structure supplied by an expert. But it should be noted that useful learning takes place even without any prespecified type information (see Constant *et al.* (1989)).

Learning and knowledge refinement occur whenever the assertion or retraction of a cue by the expert, i.e., its presentation as a positive or negative example, triggers the process of knowledge base reorganization. Since knowledge base reorganization is a costly process, it is only performed at times when there are no other demands on the system; in practice, this amounts to storing expert-supplied cues (e-cues, for short) until the expert puts LEW explicitly into knowledge base reorganization mode.

In knowledge base reorganization mode, each e-cue can bring about a complete reorganization of memory according

[2]Two environments are *identical* if they are either literally the same strings of words or they are unifiable first-order formulae.

to the following algorithm:

• Compute minimal environments. (This process of computing minimal environments is described in detail in Sect. 5.)

• Use negative examples to specialize. Negative examples serve to prune overgeneralization. If e-cue is presented as a negative example that does not yet occur in the knowledge base, it is put on a list of rejected cues that is consulted by LEW to prevent wrong generalizations. If e-cue is presented as a negative example, but is already asserted in the knowledge base, it is removed from its similarity cluster. This cluster is then restructured from scratch (see below).

• Determine the extent of the memory reorganization required by a positive e-cue.

— Introduce e-cue in as many different similarity clusters as possible. Thus e-cue is put in each cluster containing at least one cue that has a question or an answer in common with e-cue or is synonymous with it or belongs to the same type.

— Collapse the modified similarity clusters into one cluster. The clusters modified by the recent introduction of e-cue have, of course, e-cue in common and constitute by definition a unique larger cluster. This new cluster is the only one affected by the next step.

• Restructure the new cluster from scratch.

— First, forget all acquired synonyms and types. Convert all cues in S to noncompact form by replacing all synonyms and types with their values.

— Second, construct new synonyms. Each cue in the unpacked cluster S is compared with all others, and all possible synonyms are created. The propagation of the new synonyms throughout S constitutes one form of learning in LEW.

— Third, construct new types. Type construction constitutes another form of leaning in LEW. The overriding concern in type construction is to obtain a maximally compact cluster, i.e., one that both has a minimal number of cues and preserves all prior knowledge. Such a cluster is determined as follows: different knowledge-preserving clusters will result, depending on what types are built on what cues. From these temporarily created clusters LEW first selects those that have a minimal number of cues. From this set, it then selects a cluster wose effective creation leads to the construction of a minimal number of new types. This choice criterion has the double effect of condensing the knowledge base and of reducing the number of cues that are submitted to the expert for validation.

The above steps are repeated for all intermediate results of generalization until no further changes happen.

We shall now illustrate the operation of the basic LEW algorithm with a simplified application of LEW to the QUIZ advisor domain. Suppose that the following two cues are entered:

c_1: how do i report a subtotal? REPORT ITEM ⟨name⟩ SUBTOTAL!

c_2 how do i report an average? REPORT ITEM ⟨name⟩ AVERAGE!

These two cues form a cluster C_1 because they differ by the type (⟨subtotal, SUBTOTAL⟩, ⟨average, AVERAGE⟩). Suppose there exists another cluster C_2 consisting of a single cue:

c_3: how do i write a maximum? WRITE ITEM ⟨name⟩ MAXIMUM!

c_3 does not belong to C_1 because it differs from the cues in C_1 by a triple rather than by a pair. Suppose that the expert enters another cue,

c_4: how do i write an average? WRITE ITEM ⟨name⟩ AVERAGE!

We can observe that c_4 differs by a type from c_3 and by a different type from c_2. We therefore obtain two modified clusters: C_1', consisting of c_1, c_2, and c_4; and C_2', consisting of c_3 and c_4.

Since C_1' and C_2' have a common cue c_4, they collapse into one cluster C. This cluster is restructured from scratch by propagating all previous synonyms and types and thereby, in effect, forgetting them. This process restructures C into a cluster consisting of the previous cues c_1, ..., c_4, and two new cues:

c_5: how do i report a maximum? REPORT ITEM ⟨name⟩ MAXIMUM!

c_6: how do i write a subtotal? REPORT ITEM ⟨name⟩ SUBTOTAL!

C gives rise to a single synonym, i.e., (write, report), which is now propagated throughout C, giving three new cues. Similarly, the previous two types in C are replaced by the single, more comprehensive type (⟨subtotal, average, maximum⟩, ⟨SUBTOTAL, AVERAGE, MAXIMUM⟩).

After the reorganization of the above toy-sized knowledge base is completed, LEW is capable of answering a few questions, answers to which were not included in the original knowledge base, e.g., it can now answer "how do i report a maximum?"

The result of learning is now internally represented as a single, maximally compact cluster that consists of a single cue that preserves all the explicitly given information, i.e.,

how do i syn_1 a type_1? SYN_2 ITEM ⟨name⟩ TYPE_2!

where syn_1, SYN_2 belongs to the synonym (report, write) and type_1, TYPE_2 belongs to the type (⟨subtotal, average, maximum⟩, ⟨SUBTOTAL AVERAGE, MAXIMUM⟩).

5. Plan learning

To learn general sequences of actions, i.e., general plans, as quickly as possible, LEW determines for each basic action A which parts of the description of its Input_Environment and Output_Environment (E^i_A and E^o_A for short) are *relevant* to performing the action. Since (a part of) an environment description is irrelevant to the execution of an action if it is the same in E^i_A and E^o_A, the following formula is used to compute the relevant, i.e., minimal Input_Environment, min_E^i_A, for each action A:

$$\text{min\_}E^i_A = E^i_A - [E^i_A \cap E^o_A].$$

Similarly,

$$\text{min\_}E^o_A = E^o_A - [E^i_A \cap E^o_A]$$

For example[3], in the case of the action

action(1, [cube(A) ontop cube(B), cube(B) ontop table],

[3]In these examples, cube(A) denotes a logic term with a variable A, and cube(a) denotes a fully instantiated logic term.

that the IBP method can perform well on natural domains. In the presence of noise, we demonstated that IBP yields results equivalent to that of linear regression, but without having to make ad hoc assumptions. Moreover, the technique is incremental. In short, IBP is a simple, general, efficient technique which yields high quality predictions. As we have illustrated, the quality of predictions by IBP can be improved if one has appropriately weighted features. However, finding computationally efficient means for calculating these weights requires further research.

One of the criticisms of IBL techniques is that they have high storage requirements. Recent results, however, suggest that simple learning algorithms can be applied to substantially reduce storage requirements. Bradshaw's (1987) disjunctive-spanning algorithm uses an averaging technique in an effort to reduce storage requirements while maintaining high classification accuracies. Kurtzberg (1987) described an algorithm that, when trained on 288 instances (four copies of 72 handwritten symbols), saved only 121 instances and still achieved a 99.0% classification accuracy on a same-sized, disjoint set of test instances. Kibler and Aha (1987) showed that the same algorithm, when applied to Quinlan's (1987) hypothyroid disease data, saved an average of only 10 of the 220 training instances and still achieved a 97% accuracy on a disjoint set of 500 test instances. Kibler and Aha (1988) have also shown that the upper bound on the number of instances required for approximating concepts is proportional to the lengths of the boundaries separating concepts. IBL techniques do not appear to have large storage requirements for symbolic classification tasks. We anticipate that similar storage-saving techniques can be used to assist numeric value-prediction tasks.

A more severe criticism of instance-based representations is that they do not yield concise encapsulations of the data that can easily be understood and reasoned about by humans or machines. For example, BACON (Langley 1981) discovers the ideal gas low ($pV/nT = c$, where c is a constant) given numerically valued data. IBP techniques cannot represent, let alone find, this relationship. BACON heuristically searches through a space of functional expressions to find this formula. Similarly, linear regression methods search through the space of linear equations to find an easily understood relationship between the independent and dependent attributes. IBP does not yield comprehensible summaries of the data. The compensating value of IBP is that it does not require that the data satisfy some predefined model.

AHA, D.W., and KIBLER, D. 1989. Noise-tolerant instance-based learning algorithms. Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, Detroit, MI. Morgan Kaufmann, San Mateo, CA. To appear.

BRADSHAW, G. 1987. Learning about speech sounds. The NEXUS project. Proceedings of the Fourth International Workshop on Machine Learning, Irvine, CA, pp.1–11. Morgan Kaufmann, San Mateo, CA.

CONNELL, M.E., and UTGOFF, P.E. 1987. Learning to control a dynamic physical system. Proceedings of the Sixth National Conference on Artificial Intelligence. Seattle, WA, pp. 456–460. Morgan Kaufmann, San Mateo, CA.

EIN-DOR, P., and FELDMESSER, J. 1987. Attributes of the performance of central processing units: a relative performance prediction model. Communications of the ACM, **30**: 308–317.

KIBLER, D., and AHA, D.W. 1987. Learning representative exemplars of concepts: an initial case study. Proceedings of the Fourth International Workshop on Machine Learning, Irvine, CA, pp. 24–30. Morgan Kaufmann, San Mateo, CA.

——— 1988. Comparing instance-averaging with instance-filtering learning algorithms. Proceedings of the Third European Working Session on Learning, Glasgow, Scotland, pp. 63–80. Pitman Publishing, London, England.

KURTZBERG, J.M. 1987. Feature analysis for symbol recognition by elastic matching. IBM Journal of Research and Development, **31**: 91–95.

LANGLEY, P. 1981. Data-driven discovery of physical laws. Cognitive Science, **5**: 31–54.

MICHALSKI, R.S., MOZETIC, I., HONG, J., and LAVRAC, N. 1986. The multipurpose incremental learning system AQ15 and its testing application to three medical domains. Proceedings of the Fifth National Conference on Artificial Intelligence, Philadelphia, PA, pp. 1041–1045. Morgan Kaufmann, San Mateo, CA.

MITCHELL, T.M. 1982. Generalization as search. Artificial Intelligence, **18**: 203–226.

QUINLAN, J.R. 1987. A case study of inductive knowledge acquisition. *In* Applications of expert systems. *Edited by* J.R. Quinlan. Addison Wesley, Sydney, Australia.

SCHLIMMER, J.C. 1987. Incremental adjustment of representations for learning. Proceedings of the Fourth International Workshop on Machine Learning, Irvine, CA, pp. 79–90. Morgan Kaufmann, San Mateo, CA.

STANFILL, C. 1987. Memory-based reasoning applied to English pronunciation. Proceedings of the Sixth National Conference on Artificial Intelligence. Seattle, WA, pp. 577–581. Morgan Kaufmann, San Mateo, CA.

STANFILL, C., and WALTZ, D. 1986. Toward memory-based reasoning. Communications of the ACM, **29**: 1213–1228.

VALIANT, L.G. 1984. A theory of the learnable. Communications of the ACM, **27**: 1134–1142.

VERE, S.A. 1980. Multilevel counterfactuals for generalizations of relational concepts and productions. Artificial Intelligence, **14**: 139–164.

WARD 1985. 1985 model import car and truck specifications. *In* Ward's automotive yearbook, Vol. 47. *Edited by* H.A. Stark. Ward's Communication, Inc., Detroit, MI. pp. 160–163.