Instance-Based Prediction of Real-Valued Attributes

Dennis Kibler
David W. Aha
Irvine Computational Intelligence Project
Department of Information and Computer Science
University of California, Irvine, CA 92717

Technical Report 88-07

24 March 1988

To appear in *Proceedings of the 1988 Canadian Artificial Intelligence Conference.*

Instance-Based Prediction of Real-Valued Attributes

Dennis Kibler David W. Aha

Department of Information & Computer Science
University of California, Irvine
Irvine, CA 92717 U.S.A.
kibler@ics.uci.edu aha@ics.uci.edu

Abstract

Instance-based representations have been applied to numerous classification tasks with a fair amount of success. These tasks predict a symbolic class based on observed attributes. This paper presents a method for predicting a *numeric* value based on observed attributes. We prove that 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 standard approaches for value-prediction. The approach requires no background knowledge.

Keywords: incremental learning, prediction, instance-based, continuous functions

1 Introduction

Instance-based learning (IBL) strategies represent concepts using sets of instances and a similarity metric, 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 & Waltz, 1986; Stanfill, 1987), handwritten symbol recognition (Kurtzberg, 1987), thyroid disease diagnosis (Kibler & Aha, 1987), and the cart-pole balancing problem (Connell & 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. Connell and Utgoff, however, 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 method for applying instance-based techniques for predicting numeric values. Theoretical and empirical arguments are supplied to support our claims.

Most of the published work on learning from examples tasks involved the prediction of symbolic values (Vere, 1980; Mitchell, 1982; Quinlan, Compton, Horn, & Lazurus, 1986; Michalski, Mozetic, Hong, & Lavrac, 1986). In many cases, numeric attribute domains were transformed to symbolic domains in order 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 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 incremental processing of training instances, they are highly tolerant of noise in the predictor attributes (Stanfill, 1987), and they tolerate irrelevant attributes (Kibler & Aha, 1987; Stanfill, 1987). They also are good methods for numeric value-prediction tasks. We can prove that instance-based prediction (IBP) techniques are correct when given noise-free instances (see Section 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 which have differentiable functions as their solutions. Our estimation of physical processes, and in particular, the movement and equilibrium of bodies, is required in such daily events as balancing, walking, eating, pouring, cooking, driving, throwing, catching, etc. 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 light objects which appear to be heavy, can yield calamitous results.

In Section 2 we illustrate our algorithms and compare IBP techniques to linear regression techniques for prediction. 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 Section 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 by Aha & Kibler (1988).)

Section 3 summarizes our theoretical justifications for applying these techniques to numeric value-prediction tasks. We provide empirical justification in Section 4. We conclude in Section 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 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 $\mathbf{F}$ 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

$$\mathbf{F}(\langle x_1, \ldots, x_n \rangle) = C_i$$
$$\text{where} \quad \forall i \ (1 \leq i \leq n)\{x_i \in V_i\}, \text{ and}$$
$$C_i \in C.$$

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 using a set of instances and a similarity metric. A new instance is classified by some form of best-match with existing concepts.

For example, we (Kibler & Aha, 1987) have applied several IBL algorithms to two of Quinlan's thyroid disease databases (Quinlan et al, 1986). 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 set*, which is also a set of instances. The concept set 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 $\mathbf{F}(t) = \mathbf{F}(n)$, where n is t's nearest neighbor in the concept set. We measured each algorithm's performance by recording the percentage of test instances that were classified correctly by the concept set. In effect, a *concept description* (concept set plus similarity metric) described each algorithm's guess of $\mathbf{F}$ with respect to the instances on which it was trained.

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 $\mathbf{F}$ for predicting numeric domain values as follows:

$$\mathbf{F}(\langle x_1, \ldots, x_n \rangle) = r_i$$
$$\text{where} \quad \forall i \ (1 \leq i \leq n)\{x_i \in V_i\}, \text{ and}$$
$$r_i \in R.$$

$\forall t \in$ Training Set:

- $C \leftarrow C \cup \{\text{normalize}(t)\}$

$\forall t \leftarrow \text{normalize}(t'), t' \in$ Test Set:

- $\forall c \in C\{c \neq t\}$: calculate Similarity(t, c)
- Let $Sim \subseteq C$ be the set of N% most similar instances of C to t.
- Let $Sum = \sum_{c \in \text{Sim}} \text{Similarity}(t, c)$
- Then $\mathbf{F}$-estimate$(t) = \sum_{c \in \text{Sim}} \frac{\text{Similarity}(t, c)}{Sum} \times \mathbf{F}(c)$

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

$$\text{Similarity}(t, c) = \sum_{i=1}^{n} \text{Simil}(i(t), i(c))$$
$$\text{where} \quad \text{Simil}(x, y) = 1.0 - |x - y|, \text{ and}$$
$$i(t) \text{ yields the attribute value of instance } t \text{ in dimension } i$$

Table 2: Similarity of two normalised instances (n dimensions).

Connell and Utgoff (1987) recently applied IBP techniques to the cart-pole balancing problem. Their system predicted state desirability, which was continuous from -1 (undesirable) to 1 (most desirable). Saved instances had values of either -1 or 1. All other instances' degree of desirability were 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, ensuring that all attributes are assigned equal classification salience. Assuming that each attribute counts equally is not necessarily correct, 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 $\mathbf{F}(t)$ 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 the experiments is a variant of Shepard's function (Connell & 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 $\mathbf{F}$ 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.

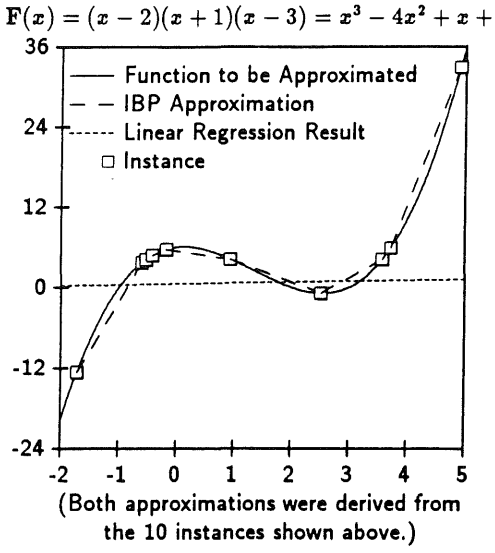

Figure 1: Approximating a typical non-linear function.

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 Figure 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 IBP methods are applicable and linear regression methods are not.

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.

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. To do this we first establish that a sufficiently large random sample of values presents a good sampling of instances. Then we demonstrate that, given a good sample of instances and values, a piecewise linear function can be generated which will closely approximate the unknown function.

[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.

3.1 Coverage Lemma

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

We start by reviewing some basic material from advanced calculus. An *ϵ-ball* about a point x of the real line R is the set of points $\{y : |y - x| < \epsilon\}$. (Note that the size of an ϵ-ball in 1 dimensional space is approximately 2ϵ.) The same definition is valid in n-dimensional Euclidean space R^n, if we interpret $|x - y|$ as the distance function in R^n.

Definition: Let X be a subset of an m-dimensional space. A subset S of X is an *ϵ-net* for X if, for all x in X there exists an s in S such that $|s - x| < \epsilon$.

We now prove that a sufficiently large random sample from the unit interval will probably be an ϵ-net.

Lemma 1. *Let ϵ and δ be fixed positive numbers less than 1. A random sample S containing $m > 1/\epsilon \times ln(1/\epsilon\delta)$ instances from $[0,1]$ will form an ϵ-net with confidence greater than $1 - \delta$.*

Proof: We prove this lemma by partitioning the unit interval into k equal-sized sub-intervals, each with size less than ϵ. We also ensure that, with high probability, at least one of the m sample instances lies in each sub-interval.

Let $k > 1/\epsilon$. The probability that arbitrary instance $i \in [0,1]$ will lie in some selected sub-interval is $1 - 1/k$. The probability that none of the m sample instances will lie in a selected sub-interval is $(1 - 1/k)^m$. The probability that any sub-interval is excluded by all m sample instances is $k \times (1 - 1/k)^m$. Since $(1 - 1/k)^m < e^{-m/k}$, then

$$k \times (1 - 1/k)^m < k \times e^{-m/k}.$$

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

$$\begin{aligned} k \times e^{-m/k} &< \delta \\ e^{-m/k} &< \delta/k \\ -m/k &< ln(\delta/k) \\ m &> -k \times ln(\delta/k) \\ m &> k \times ln(k/\delta) \end{aligned}$$

Consequently, with probability greater than $1-\delta$, each sub-interval contains some sample instance of S. Since each instance of $[0,1]$ is in some sub-interval, then, with this same probability, an arbitrary instance of I is within ϵ of some instance of S. ■

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

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.

Definition: A function f is an *ϵ-approximation* of a function g if they have the same domain, and for all instances x in their common domain $|f(x) - g(x)| < \epsilon$.

The following definition is motivated by Valiant's (1984) work on learnability theory.

Definition: A function f is *learnable by IBP techniques* when, for $1 > \delta, \epsilon > 0$, the function $\tilde{f}$ generated by IBP is an ϵ-approximation of f with probability greater than $1-\delta$.

Note that our definition does not specify how long learning might take, only that it usually converges to approximately the right answer.

Definition: The slope of a function f is *bounded on X by B* if

$$\forall x, x' \in X \ (x \neq x') \ \left|\frac{f(x) - f(x')}{x - x'}\right| \leq B.$$

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

Theorem 1. *Let f be a continuous, real-valued function on $[0,1]$ with slope bounded by B. Then f is learnable by IBP techniques.*

Proof: Let f be a continuous function on $[0,1]$. Let 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 on the slope of f. In particular, we apply Lemma 1 with $\epsilon' = \epsilon/2B$. (We assume, without loss of generality, that $B \geq 1$.) The lemma guarantees that, if m is large enough, then we will have an $\epsilon/2B$-net for $[0,1]$ with confidence greater than $1-\delta$. More specifically, we let S be a random sample of $[0,1]$ with size $m > 2B/\epsilon \times ln(2B/\epsilon\delta)$.

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

Now we must show that $\tilde{f}$ is an ϵ-approximation of f. Let x be an arbitrary point in $[0,1]$. We shall consider first the case where x is surrounded by sample instances x' and x'' in S, where x' is the nearest neighbor of x in S. Note that

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

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

Since B is the upper bound on the slope between any two points for $f(x)$, it is also an upper bound on the slope between any two points for $\tilde{f}(x)$. Therefore,

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

and

$$|\tilde{f}(x) - \tilde{f}(x')| = |\tilde{s}(x', x'') \times (x - x')| < B \times \epsilon/2B = \epsilon/2.$$

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

$$|f(x) - \tilde{f}(x)| < \epsilon/2 + 0 + \epsilon/2 = \epsilon.$$

The second case, where x is not surrounded by sample instances in S, can be handled similarly. This proves that the piecewise linear approximation $\tilde{f}$ is an ϵ-approximation of f. ■

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

Theorem 2. *If f is a continuous function on a closed and bounded n-dimensional space with derivative bounded by B, then f is learnable by IBP techniques. In particular, for given positive values of ϵ and δ, $m > 2B/\epsilon^n \times ln(2B/\epsilon^n\delta)$ samples will suffice.*

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 in order to ensure that the time to learn is polynomially bounded. In particular, we compensate for looser constraints on the domain of the function by tightening the constraints on how fast the function can change its value over a small interval. That is, we require that the derivative of the function exist and be uniformly bounded.

The requirement for the bound on the derivative is illustrated by the function $sin(1/x)$ on the open interval (0,1]. Note that as x approaches 0, the derivative (slope) of the function is unbounded. As is easily seen, for any ϵ between 0 and 1, there is no piecewise linear ϵ-approximation of this function.

While these results establish the appropriateness of IBP for noise-free functions, real world data requires attention to noise. A standard method for tolerating noise is to use some form of averaging. In particular, we believe that the following change to the algorithm yields a method which works for noise in the function value, but the proof eludes us.

Instead of constructing a piecewise linear approximation based on a single ϵ-net, consider forming m ϵ-nets, each net yielding a piecewise linear approximation $\tilde{f}_i$. We define $\tilde{f}$ to be the (pointwise) average of these $\tilde{f}_i$. Clearly $\tilde{f}$ is still piecewise linear. We believe that, the resulting $\tilde{f}$ will be a good approximation. We demonstrate the appropriateness of this method with an empirical study.

4 Empirical Justification

In this section we describe our experiments and show how the results support our conjecture. In particular, we note that instance-based learning techniques achieve the same accuracy as linear regression methods. We note that if the underlying function was exactly linear then both linear regression and IBP would yield perfect predictions. The value of IBP is that it requires that the function be only locally linear while linear regression requires that the function be globally linear. Also, IBP requires no *ad hoc* adjustments to the underlying method, which are often made when applying linear regression models.

Acronym	Attribute Name	Unit
MCYT	Machine Cycle Time	Nanoseconds
MMIN	Minimum Main Memory	Kilobytes
MMAX	Maximum Main Memory	Kilobytes
CACH	Cache Memory Size	Kilobytes
CHMIN	Min Number of I/O Channels	Channels
CHMAX	Max Number of I/O Channels	Channels

Table 3: Predictive attributes of the central processing units database.

Vendor/Model	MCYT CHMIN	MMIN CHMAX	MMAX	CACH PRP
IBM 4341-12	185 1	2000 6	16000	16 76
DEC-Vax-11/750	320 1	512 5	8000	4 40
Sperry 1100/94	30 12	8000 176	64000	128 1150

Table 4: Example instances of the CPU characteristics database.

4.1 Experimental Data

We chose to experiment with the database published by Ein-Dor and Feldmesser (1987) in the April 1987 issue of the Communications of the ACM. The authors described a stepwise multivariate linear regression technique for predicting the published relative performance (PRP) of central processing units. Each of the 209 cpu data instances is represented with six predictive attributes (described in Table 3). The other attributes included the vendor, the model name, and its published relative performance. Three example instances are shown in Table 4.

The purpose of their article was to describe a predictive model for computer systems that was simpler than queueing networks. Our purpose here is to demonstrate that IBP techniques can be used to describe models of equal simplicity and accuracy for approximating the published relative performance function.

We chose to experiment with this data set for several reasons:

1. The authors presented a case study that allowed us to contrast a linear regression model with an IBP model on the same data set.
2. We wanted to show that IBP models support predictive behavior for natural databases (as opposed to being restricted to carefully crafted artificial databases).
3. We were interested in investigating the predictive quality of IBP techniques when both the input and output attributes were numeric.

Range of Relative Performance	Number of Instances	% Average Deviation		
		Linear Regression	IBP	
			Raw	Tuned
0–20	31	72.17	81.38	55.62
21–100	121	28.64	27.88	29.64
101–200	27	28.57	27.91	31.60
201–300	13	23.93	19.16	22.21
301–400	7	21.49	22.12	27.14
401–500	4	18.72	16.80	16.20
501–600	2	17.35	11.86	30.49
600+	4	10.34	43.69	33.35
All	209	33.91	35.02	33.02

Table 5: Average deviation of relative performance predictions.

4.2 Experiment and Results

We carried out two experiments with the cpu performance data. In each case, the function **F** to be predicted was PRP. The results of both our own experiments and the original linear regression experiment are displayed in Table 5. All results are in terms of the average deviation of the predicted and actual cpu performance values.[2]

Ein-Dor and Feldmesser (1987) calculated the linear regression equation for cpu performance and then used it to predict each instance's relative cpu performance. Since regression minimizes the square of the absolute error, it will appear to be more accurate for large values when evaluated in terms of relative error.

In our first experiment (see column Raw in Table 5), each instance was described in terms of the 6 given attributes. The training and test sets were identical; they contained all 209 instances. We set the value for N to 3%, meaning that the IBP prediction of a test instance's relative cpu performance was based on its 3% most similar instances in the concept set (excluding the test instance itself). The proximity algorithm fares relatively well in each range except the first and last. The values of the first (smallest-valued) range were sensitive to small absolute errors and thus produced the greatest relative error. The last range contains the highest values, which were few in number and highly scattered. Actually it would be easy to assign some measure of confidence with IBP which would suggest that the predicted value was tenuous, at best, for the high range.

The overall average deviation of predicted and actual values for the first experiment is surprisingly similar to that of the linear regression experiment, which was only slightly better.

What differed between the experiments is of greater importance. The regression experiment required a great deal of application-dependent knowledge while the first IBP experiment employed none. Ein-Dor and Feldmesser analyzed the data and subject domain and reviewed the linear correlations of the PRP values with the 6 independent attributes. They concluded that transformations of the independent

[2]The published value for overall average deviation of the linear regression model is 34.10 (Ein-Dor & Feldmesser, 1987), which disagrees with our calculation of 33.91. Otherwise, our calculations are in agreement.

Derived Attribute	Attribute Transformation
Average Memory Size	$(\text{MMIN} + \text{MMAX})/2 \times 10^{-3}$
Cache Memory Size	$\text{CACH} \times 10^{-1}$ (or 0 if none)
Channel Capacity	$(\text{CHMIN} + \text{CHMAX})/2 + 1$ $\times(1/\text{MYCT}) \times 10$

Table 6: Transformations of the 6 independent attributes.

and dependent attributes were needed to enhance the regression model's predictiveness. First, they chose to employ a square-root transformation of the dependent attribute. They then chose to employ 3 "tuned" attributes to be used as independent attributes for prediction of the square-root of the published relative performance values. The tuned attributes are average memory size, $1/10^{th}$ of the cache size, and channel capacity. The transformations are listed in Table 6.

The problem they faced is a typical one when using linear regression. What do you do when the dependent variable is not a linear combination of the attributes? In their case, by appealing to domain knowledge, specifically Grosch's law, they reasoned that the *square root* of the performance would be a linear combination of measured attributes. Finding this particular transformation requires a model of the domain. IBP does as well without introducing domain expertise.

In our second experiment (see column Tuned in Table 5), we used the same tuned input attributes for the IBP method as thosed used by the linear regression technique. This was done by calculating the values for the three tuned attributes and representing the instances in terms of them. The effect of tuning attributes is to give a better weight to each individual attribute. The average deviation results for this experiment also appear in Table 5. In this case, we find that the IBP predictions are slightly better than those given by the linear regression model, again measured in terms of average deviation.

5 Discussion

Continuous functions are an extremely large class of functions. In particular, they can represent the behavior of physical systems. Our ability to interact successfully with the world depends, in part, on our ability to predict the continuously changing environment. Given a sufficiently large sample size, we have shown that IBP is guaranteed to yield a good approximation for continuous functions. In the presence of noise, we demonstrated 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. Finding computationally efficient means for calculating these weights, however, requires further research.

One of the criticisms of IBL techniques is that they have high storage requirements. Recent results, however, have suggested that simple learning algorithms can be applied that greatly ease 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 et al's (1986) 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. We (Kibler & Aha, 1988) have 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 law ($pV/nT = c$, 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.

Acknowledgements

We would like to thank Marc Albert for suggesting a revision of Theorem 1.

References

Aha, D. W., & Kibler, D. (1988). *Detecting and removing noisy instances from concept descriptions.* Manuscript submitted for publication.

Bradshaw, G. (1987). Learning about speech sounds: The NEXUS project. In *Proceedings of the Fourth International Workshop on Machine Learning* (pp. 1–11). Irvine, CA: Morgan Kaufmann.

Connell, M. E., & Utgoff, P. E. (1987). Learning to control a dynamic physical system. In *Proceedings of the Sixth National Conference on Artificial Intelligence* (pp. 456–460). Seattle, WA: Morgan Kaufmann.

Ein-Dor, P., & Feldmesser, J. (1987). Attributes of the performance of central processing units: A relative performance prediction model. *Communications of the Association for Computing Machinery, 30*, 308–317.

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-saving learning algorithms* (Technical Report 88-06). Irvine, CA: University of California, Irvine, Department of Information and Computer Science.

Kurtzberg, J. M. (1987). Feature analysis for symbol recognition by elastic matching. *I.B.M. 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., & Lavrac, 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.

Mitchell, T. M. (1982). Generalization as search. *Artificial Intelligence, 18*, 203–226.

Quinlan, J. R., Compton, P. J., Horn, K. A., & Lazurus, L. (1986). Inductive knowledge acquisition: A case study. In *Proceedings of the Second Australian Conference on Applications of Expert Systems*. Sydney, Australia.

Schlimmer, J. C. (1987). Incremental adjustment of representations for learning. In *Proceedings of the Fourth International Workshop on Machine Learning* (pp. 79–90). Irvine, CA: Morgan Kaufmann.

Stanfill, C., & Waltz, D. (1986). Toward memory-based reasoning. *Communications of the ACM, 29*, 1213–1228.

Stanfill, C. (1987). Memory-based reasoning applied to English pronunciation. In *Proceedings of the Sixth National Conference on Artificial Intelligence* (pp. 577–581). Seattle, WA: Morgan Kaufmann.

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

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