AN EFFICIENT METRIC FOR HETEROGENEOUS INDUCTIVE LEARNING APPLICATIONS IN THE ATTRIBUTE-VALUE LANGUAGE[1]

Christophe Giraud-Carrier and Tony Martinez
Brigham Young University, Department of Computer Science, Provo, UT 84602

Abstract. Many inductive learning problems can be expressed in the classical attribute-value language. In order to learn and to generalize, learning systems often rely on some measure of similarity between their current knowledge base and new information. The attribute-value language defines a heterogeneous multi-dimensional input space, where some attributes are nominal and others linear. Defining similarity, or proximity, of two points in such input spaces is non trivial. We discuss two representative homogeneous metrics and show examples of why they are limited to their own domains. We then address the issues raised by the design of a heterogeneous metric for inductive learning systems. In particular, we discuss the need for normalization and the impact of don't-care values. We propose a heterogeneous metric and evaluate it empirically on a simplified version of ILA.

1. Introduction

Many inductive learning systems use the classical attribute-value language as their representation language. In the attribute-value language, a training example is a vector whose entries are pairs consisting of an attribute name and its value. In most cases, attribute names are omitted as the context unambiguously determines which entry corresponds to which attribute. Some of the attributes are identified as being input attributes, and some (often a single one) are identified as output attributes. Each input attribute's value ranges over some domain, and the cross product of all the domains defines the input space of the application. Hence, training examples are points in the input space, labeled with some output value.

In many cases, inductive learners are concerned with the classification of points into disjoint categories. The system uses training examples to derive a classification that is then generalized to previously unseen points of the input space. Generalization typically consists of either one (or a combination) of two processes: 1) the discovery of subsets of points (or regions) of the input space that map to a given concept [4, 5], and 2) the use of proximity (or similarity) to known points [2, 6, 7, 12]. When a new point is shown, its classification is determined either by its belonging to one of the identified regions, or by the classification of the "closest" known points. Classification by closeness is the essence of most nearest-neighbor algorithms (see [3] for a survey) and memory-based reasoning (see for example [11]).

To find the "closest" known points to a new point in the input space, the system needs some measure of proximity or similarity, i.e., a distance metric defined on the input space. Because the domains of attributes vary in nature, the input space is often heterogeneous, in the sense that topological properties such as distance do not have the same definition in all dimensions. The notion of distance between two points in such heterogeneous spaces is non trivial. This paper shows the limitations of homogeneous metrics, and proposes a (combined) heterogeneous distance measure that accounts for heterogeneous spaces, in the context of inductive learning. Empirical results demonstrate the superiority of the proposed measure over homogeneous metrics.

---

[1]This work was supported in part by grants from Novell Inc., and WorPerfect Corp.

Section 2 addresses the issue of similarity, discusses two representative homogeneous metrics, and gives examples of the inadequacy of these metrics in heterogeneous domains. Section 3 discusses two issues raised by the design of a metric for heterogeneous inductive learning applications. Section 4 proposes a heterogeneous metric that extends the similarity function of [2]. Section 5 overviews a simplified version of ILA [6] that serves as a basis for empirical evaluation. Finally, Section 6 concludes the paper.

2. Considerations on the Notion of Similarity

We consider here two types of attributes: nominal and linear. Nominal attributes have discrete values that are not related in any way other than the fact that they belong to the same set. Linear attributes have values that can be ordered in the usual sense, and whose ordering is relevant to the context in which they are used. Linear attributes may be discrete and ordered or continuous. The attribute *blood group*, for example, is a nominal attribute, while the attribute *weight* is linear. Each kind of attribute gives rise to a distinct notion of similarity or distance. We discuss two of them here, and show that they are mutually incompatible. The selected metrics are in no wise unique, only representative. They serve as illustration, are commonly used [2, 5, 12], and have been found to give good empirical results on their respective domains of applications.

2.1. Distance for Nominal Spaces

For nominal data, the notion of how far apart two values are reduces to a simple binary relation. Two values are either the same, or they are different. If the values are the same, then the distance is 0; otherwise, the distance is 1. The one-dimensional definition extends to the multi-dimensional case in a straight forward way. Formally,

$$\textit{Let } \boldsymbol{x} = (x_i) \textit{ and } \boldsymbol{y} = (y_i) \textit{ be two } n\textit{-dimensional nominal vectors}$$

$$\textit{Let } dn(x_i, y_i) = \begin{cases} 1 & \textit{if } x_i \neq y_i \\ 0 & \textit{otherwise} \end{cases}$$

$$\textit{Then } DN(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{n} dn(x_i, y_i)$$

DN (Distance for Nominal spaces) conveys the intuitive idea that the farther away two vectors are, the less similar they are. So it can be used directly to choose a closest match.

However, *DN* is inadequate on linear spaces. Consider the following example, where each attribute is linear and ranges over [0, 10] in the natural numbers.

$$x = (2,1,3)$$
$$y = (3,1,4) \Rightarrow DN(x,y) = 2$$
$$z = (2,6,0) \Rightarrow DN(x,z) = 2$$

Though $DN(x,y)=DN(x,z)$, it appears that (given linear attributes) y is closer to x than z is. The problem is that the ordering implies varied magnitudes in the differences between values, while *DN* only accounts for equality or inequality. It takes any magnitude greater than 0 to 1, and leaves 0 magnitudes at 0. In some sense, it forces a step function on a linear domain. Things become even worse in continuous spaces where the probability of any two values being equal is extremely low.

2.2. DISTANCE FOR LINEAR SPACES

Linear attributes are ordered (e.g., continuous values). The ordering gives rise to a natural measure of distance, namely, the farther away things are in the ordering, the larger their distance should be. This is consistent with our everyday notion of distance. The distance between two values can be defined as the absolute value of their difference. Since every ordered set is in one-to-one correspondence with a subset of the natural numbers, this distance is well-defined on all domains. A common generalization of this one-dimensional definition to the multi-dimensional case is the classical Euclidean distance.

However, this definition implicitly assumes that all attributes range over the same domain (e.g., reals, integers). If the domains are different, then some attribute distances may dominate others in the overall distance. For example, if x and y are linear attributes ranging over [0, 1] in the real numbers and [0, 100] in the natural numbers, respectively, then distances along y are likely to dominate distances along x. The smallest distance in y is equal to the largest distance in x. The problem is one of scale. We thus modify the definition as follows.

First, let us argue that in most practical learning applications, linear attributes are bounded, that is they have a smallest and a largest possible value. These can easily be obtained from the associated dataset. To eliminate the effects of statistical outliers, the dataset must be ridden of examples whose attributes have such "irregular" values. Alternatively, the linear attributes (especially continuous ones) can be discretized into finitely many classes.

Then, let *range(i)* denote the range of values for attribute i, i.e., the difference between the maximum and minimum values of attribute i. We now define *NDL* (Normalized Distance for Linear spaces) by:

$$\begin{aligned}
&\textit{Let } \mathbf{x} = (x_i) \textit{ and } \mathbf{y} = (y_i) \textit{ be two } n\textit{-dimensional linear vectors} \\
&\textit{Let } ndl(x_i, y_i) = \frac{|x_i - y_i|}{range(i)} \\
&\textit{Then } NDL(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{n} [ndl(x_i, y_i)]^2}
\end{aligned}$$

The division by *range(i)* causes all attribute distances to fall within the range [0, 1]. Hence, all attributes make a normalized contribution to *NDL*. Again, *NDL* conveys the intuitive idea that the farther away two vectors are, the less similar they are. So it can be used directly to choose a closest match.

However, *NDL* is inadequate on nominal spaces. Consider the following example, where each attribute ranges over the discrete set {0, 1, 2, 3, 4, 5}. For nominal domains, we let *range(i)* be the number of possible values in the domain minus 1.

$$\begin{aligned}
&x = (3,1,2) \\
&y = (2,1,1) \Rightarrow NDL(x,y) = \sqrt{2}/5 \\
&z = (3,4,2) \Rightarrow NDL(x,z) = 3/5
\end{aligned}$$

Though $NDL(x,y) < NDL(x,z)$, it appears that (given nominal attributes) z is closer to x than y is. The problem is that *NDL* treats nominal values as if they had inherent magnitudes. Instead of making the distance between any two distinct nominal values the same, it assigns different distances based on some non-existent, but assumed, intrinsic ordering.

One may be tempted to claim that the above example is a result of an artifact of encoding. If all nominal values are "one-hot" encoded (i.e., as many bits as there are values, one set for each distinct value), then *NDL* would make z closer to x as expected. However, such encodings cause a multiplication of the number of attributes. Moreover, they reduce nominal spaces to Boolean spaces, and it is straight forward to show that on Boolean spaces *NDL=DN*.

2.3. Inadequacy of Homogeneous Metrics on Heterogeneous Spaces

The above examples show that if a learning system uses a homogeneous metric and applies it in a non-discriminatory way to all applications, then it is likely that its accuracy will suffer on some applications. It is clear that similar results hold when applying either of the homogeneous metrics to a heterogeneous space. Consider the following example, where the first attribute is nominal and ranges over the discrete set {0, 1, 2, 3, 4, 5}, and the other two attributes are linear and range over [0, 10] in the natural numbers.

$$x = (4,1,0)$$
$$y = (4,7,0) \Rightarrow NDL(x,y) = 6/10 \wedge DN(x,y) = 1$$
$$z = (3,0,2) \Rightarrow NDL(x,\ z) = \sqrt{6}/10 \wedge DN(x,z) = 3$$

The two distances naturally give inconsistent results. *DN* leads to the conclusion that y is closer to x than z is, while *NDL* leads to the opposite conclusion. This inconsistency is a direct consequence of the problems discussed above. It follows that to correctly handle heterogeneous spaces, a heterogeneous metric must be employed.

3. Designing a Metric for Heterogeneous Learning Applications

We address two important issues raised by the design of a metric for heterogeneous learning applications. One has to do with the heterogeneity of the learning space, and the other with the common use of rules containing *don't-care* values.

3.1. Heterogeneity of the Learning Space

In designing a metric for heterogeneous spaces, it is desirable not to loose accuracy on homogeneous spaces. One simple solution consists of combining the homogeneous metrics into a single metric that applies each homogeneous metric to its corresponding attributes. Such a combination may suffer from the scale problem.

In this case, the problem is no longer with each attribute's range, but with the ranges of values covered by the homogeneous metrics. If these are different, then one of the metrics may indeed dominate the overall result. Hence, it is not sufficient that each metric be adequate on its own space. They must be somehow normalized with respect to each other. The normalization in *ndl* causes *ndl* to have exactly the same range of values as *dn*, namely [0, 1]. Hence, not only is *ndl* a more accurate linear metric (for linear spaces), it is also "compatible" with *dn*.

There is a possible problem with such a normalization however. Indeed, by normalizing, we artificially force complete mismatches (i.e., distance of 1) to the extrema. It would be reasonable instead to assume that when two values are far enough apart (based on some threshold value), then there is a complete mismatch between them. In other word, a non-linear normalization technique

may be more appropriate. Another alternative consists of using multiplicative scale factors on nominal attribute distances. We do not address this problem in this paper.

3.2. Using Rules with Don't-Care Values

A common and useful extension, made to the classical attribute-value language, is the addition of a special *don't-care* symbol for input attributes. The interpretation for a vector containing input attributes set to don't-care is as follows. An attribute whose value is not don't-care is said to be *asserted*. A vector whose input attributes are all asserted represents a single point of the input space. A vector that has input attributes set to don't-care represents all of the points in the input space that would be obtained by replacing all of the don't-care symbols by any one of the possible values of the corresponding input attributes. Consider the following example where each attribute ranges over the discrete set {1, 2, 3, 4}.

Let $\boldsymbol{v} = (1,*,3,2)$

Then $\boldsymbol{v}$ *represents the following points*

$(1,1,3,2)$, $(1,2,3,2)$, $(1,3,3,2)$, *and* $(1,4,3,2)$

Hence, such vectors represent subsets of points, or regions, in the input space. We discuss two situations in which they arise, and show the impact they have on the definitions of distance.

3.2.1. Origins of Don't-Cares. Vectors with input attributes set to don't-care represent subsets of points in the input space. Hence, they can be viewed as *rules* in the sense that they map a set of points, rather than a single point, to a given output value (or concept). The number of points represented by a rule is a measure of that rule's *generality*. The larger the set of points, the greater the generality. Such rules arise in at least two situations that are both relevant to learning systems.

The purpose of inductive learning systems is to extract, from a set of training examples, the critical features that govern an application. Some of these critical features, or set of attribute-value pairs that are good predictors of the output, may be represented as rules. Such rules are the result of the learning system's ability to generalize. A commonly used generalization rule in learning systems is the *dropping-the-condition* rule [8]. In the attribute-value language, it translates into the setting of one (or more) of the input attributes of a given vector to don't-care. The input attributes that remain asserted can then be viewed as critical features (those that are don't-care do not affect the output value).

Another (possibly complementary) situation in which rules arise is by being provided *a priori* (by some teacher). Indeed, such rules may also be used as an encoding of prior knowledge, or as a particular instantiation of domain knowledge or commonsense. They then serve as learning biases for the system (see, for example [5]). For example, when learning about the flying abilities of birds, one may bias the system with a rule encoding the fact that birds typically fly. The system then only needs to focus on exceptions.

3.2.2. Effects of Don't-Cares. Since they do not have any intrinsic value, don't-cares are not comparable with other values. To find a closest vector in the current knowledge base, distance is measured from vectors in the knowledge base to new vectors. If $\boldsymbol{x}$ and $\boldsymbol{y}$ are two vectors such that all of their attribute values are equal except for some attributes that are don't-care in $\boldsymbol{x}$ and asserted in $\boldsymbol{y}$, and if d is any well-defined distance, then we claim that the following should hold.

$$d(\boldsymbol{x},\boldsymbol{y}) = 0 \qquad (1)$$
$$d(\boldsymbol{y},\boldsymbol{x}) \neq 0 \qquad (2)$$

That is, the distance measure is not symmetric. In (1), $\boldsymbol{x}$ is in the current knowledge base, $\boldsymbol{y}$ is a new vector, and the equality is an immediate (and natural) consequence of the definition of don't-cares. In (2), $\boldsymbol{y}$ is in the current knowledge base, $\boldsymbol{x}$ is a new vector, and the inequality requires some explanation.

If reasonable at all, inequality (2) is only useful when unseen vectors (either in training or in testing) are allowed to have don't-care input attributes. We argue that such situations may arise and claim that in such cases, the inequality should hold. As seen in Section 3.2.1., rules may be the expression of domain knowledge. There is no reason to assume that all of these are available *a priori*. It is plausible that the learning system (much like a human in this case) is exposed to examples and rules in an unpredetermined order. Hence, certain rules may come in later in the learning process. For example, if you live in Australia, most of the birds you see are ostriches that do not fly. Thus, you are exposed to exceptions before you may be taught the more general rule about birds flying ability. In testing, allowing don't-cares is a way to potentially discover default rules for situations in which much information is missing. This is particularly useful in bottom-up inheritance, where, based on the current knowledge base, one can obtain the default classification of a very large set of points. In this latter case, a don't-care can also be thought of as a don't-know or missing value.

If (1) holds and the distance is symmetric, then the generality of the new vector may cause several vectors in the current knowledge base to be equidistant to it, as in the following example, where all attributes are Boolean, and the last one is the output. The knowledge base is *KB*, the new vector is $\boldsymbol{x}$, and $*$ stands for don't-care.

$$KB = \{(1,1,0,1,1),(1,1,*,1,0),(*,0,*,1,1)\}$$
$$\boldsymbol{x} = (1,*,*,0,?)$$

All of the vectors in *KB* have the same distance to $\boldsymbol{x}$. However, in this case, where a default classification is expected, it seems reasonable to expect the more general of the vectors in *KB* to be closest. Inequality (2) increases the distance to more specific existing vectors, thus causing the system to favor the more general vectors.

Another impact of the variation in generality of different vectors and (1) is the need for some normalization of the overall distance, so as to distinguish a perfect match from a don't-care. Since attributewise distances range over [0, 1], this can be done by dividing the overall distance by the number of asserted input attributes in the vector from which distance is computed. For example, it is better to have 2 mismatches out of 5 asserted input attributes than 1 mismatch out of only 2 asserted input attributes.

4. DHET - A Metric for Heterogeneous Inductive Learning

Based on the above discussion, we propose the following modified set of definitions. Let *num_asserted(**x**)* be the number of asserted input attributes in vector $\boldsymbol{x}$.

$$Let\ \boldsymbol{x} = (x_i)\ and\ \boldsymbol{y} = (y_i)\ \ be\ two\ n\text{-}dimensional\ vectors$$

$$Let\ dn'(x_i, y_i) = \begin{cases} 1 & if\ x_i \neq * \wedge x_i \neq y_i \\ 0 & otherwise \end{cases}$$

$$Then\ DN'(\boldsymbol{x}, \boldsymbol{y}) = \frac{\sum_{i=1}^{n} dn'(x_i, y_i)}{num_asserted(\boldsymbol{x})}$$

$$Let\ ndl'(x_i, y_i) = \begin{cases} \dfrac{|x_i - y_i|}{range(i)} & if\ x_i \neq * \wedge y_i \neq * \\ 1 & if\ x_i \neq * \wedge y_i = * \\ 0 & otherwise \end{cases}$$

$$Then\ NDL'(\boldsymbol{x}, \boldsymbol{y}) = \frac{\sqrt{\sum_{i=1}^{n} [ndl'(x_i, y_i)]^2}}{num_asserted(\boldsymbol{x})}$$

$$Finally,\ DHET(\boldsymbol{x}, \boldsymbol{y}) = \frac{\sqrt{\sum_{i=1}^{n} \begin{cases} dn'(x_i, y_i) & if\ attribute\ i\ is\ nominal \\ [ndl'(x_i, y_i)]^2 & if\ attribute\ i\ is\ linear \end{cases}}}{num_asserted(\boldsymbol{x})}$$

The value of 1 for *dn'* and *ndl'* when x_i is asserted and y_i is not reflects the fact that we consider such a case to be a "complete" mismatch. *DN'* is the reciprocal of the conceptual similarity measure of [12]. *DHET* (Distance for HETerogeneous spaces) is an extension of IBL's similarity function [2]. The two metrics are equivalent when a purely instance-based learning algorithm is used, and *DHET* extends IBL's similarity function to inductive learning algorithms that use and/or create general rules. *DHET* is also similar to the distance metric used by NGE [10]. It is somewhat simpler since rules containing don't-cares represent hyperplanes rather than hyperrectangles. It is also inherently heterogeneous whereas NGE's distance is designed for purely continuous domains. Extensions to heterogeneous spaces have been proposed (see for example [13]). NGE's distance is also a weighted sum, where each attributewise distance is assigned its own weight in the computation of the final distance. Mechanisms to assign weights to attributes in distance computation may be found in [10, 11, 13]. *DHET* could be similarly extended.

5. Evaluation

In this section, we give empirical evidence of the adequacy of *DHET*. The results were obtained by executing the same algorithm, only varying the metric it uses. ILA (Incremental Learning Algorithm) was chosen because of its simplicity, execution speed, and the direct impact the metric has over its accuracy. ILA bears resemblance with IBL algorithms [2] and NGE [10]. It can be viewed as a nearest-hyperplane algorithm. We first overview a simplified version of ILA. Details of the full algorithm are in [6]. We then present results of simulations on a variety of applications.

5.1. ILA - Algorithmic Overview

ILA's representation language is the classical attribute-value language, with the addition of the special *don't-care* symbol. For convenience, examples and rules are viewed as *input-output pairs*.

Examples have all of their input attributes asserted to some value. ILA discovers new rules by generalizing from examples. The generalization rule it uses is the classical drop the condition rule [8], which causes the value of an input attribute to be set to don't-care.

ILA implements a network of simple nodes that adapts to newly acquired knowledge by dynamically adding and/or removing nodes in the network. The network's structure is a balanced binary tree. No parameters need to be set *a priori*. Parallelism at the node level is inherent.

For our purposes, each node stores a pair, together with counters for each output value (vector *counters*), and a counter for the number of pairs covered (*num_covers*). The counters are used to handle noise. For each output value, they keep track of the number of training pairs whose inputs match exactly the input of the node's stored pair. The output whose count is highest is the node's output. The counter num_covers keeps track of the number of pairs that have the same output as the stored pair, and are covered by it. We say that a pair *p covers* a pair *q* if and only if the set of points represented by *q* is a proper subset of the set of points represented by *p*. The value of num_covers can serve as the degree of confidence in the node's stored rule.

Training pairs are incrementally presented to the system. ILA's learning consists of two phases: *execution* and *adaptation*. In the execution phase, the training pair is presented to the network and a winning node is identified. Identifying the winner is a simple broadcast (of the training pair) and gather operation, that first minimizes the distance (i.e., node with smallest distance wins), then resolves ties by giving priority to the more specific node (i.e., larger number of asserted inputs), then to the node with the largest value of num_covers. Winning distances are passed up and nodes are tagged as self, right or left, based on which node wins the competition. A winner is then easily identified by going down the tree until a node marked as self is found. If the network is empty, a new node is automatically added for the training pair. Giving priority to more specific nodes helps in handling exceptions, and giving priority to nodes whose num_covers value is higher increases the chances of selecting a "most likely to be correct" winner.

In the adaptation phase, counters and num_covers are updated, and a new node is created for the training pair, if 1) the training pair is an exception to, 2) covers, or 3) is too different from the winning node's stored pair. If it covers it and has the same output, then the winning node is also deleted. An attribute is dropped if the training pair and the winning node's stored pair have the same output, differ in the value of exactly one of their asserted input attributes, and their relative numbers of asserted input attributes differ by at most 1. Precautions are taken to avoid the creation of a rule that covers all of the input space (i.e., drop all attributes).

Evaluating ILA for predictive accuracy on a test set consists of presenting every pair in the test set and only running the execution phase. The output of the winning node is the system's output and prediction of the new pair's output.

5.2. Simulation Results

A variety of datasets from the Irvine repository of machine learning datasets [9] were chosen, representing a mixture of homogeneous and heterogeneous applications. Two modifications are made however. All continuous attributes are made discrete (using a K-means algorithm on each input), and all don't-know values are arbitrarily treated as don't-care values. Other (possibly more appropriate) ways of handling don't-know values are the topic of future research (three of them are discussed in [1]). Also, even though the attributes of the application lenses are all claimed to be nominal, the first one (i.e., age) appears to have a definite natural ordering to it. Hence, the dataset lenses is considered to be heterogeneous.

ILA's predictive accuracy (in %) on a test set was gathered for each application. Each of the reported results is an average over 10 runs of ILA for that application. In each run, the training set and the test set are randomly regenerated. Each set contains one half of the examples in the complete dataset. The datasets used and corresponding results of simulations with *DN'*, *NDL'*, and *DHET* are given in Table 1. Table 2 summarizes the averages on each kind of datasets and shows the overall averages for all the datasets.

Table 1 - Datasets Used and ILA's Simulation Results

Dataset	#instances	#inputs	#out. values	DN'	NDL'	DHET
zoo	90	16N	7	95.3	94.7	95.3
mushroom	8124	22N	2	100	100	100
chess-end	3196	36N	2	90.5	90.1	90.5
tic-tac-toe	958	9N	2	80.4	75.8	80.4
LED	1000	7N	10	70.2	70.2	70.2
molecular-bio	106	57N	2	77.9	66.8	77.9
iris	150	4L	3	90.5	94.0	94.0
glass	214	9L	7	66.4	67.9	67.9
breast-cancer	699	9L	2	94.4	94.7	94.7
pima-indians	768	8L	2	63.4	67.2	67.2
vowel	528	10L	11	78.2	91.9	91.9
sonar	208	60L	2	79.8	85.4	85.4
lenses	24	3N, 1L	3	65.0	80.0	80.0
hepatitis	155	13N, 6L	2	77.3	78.6	78.6
soybean-sm.	47	31N, 4L	4	98.8	98.8	98.8
credit-screen.	690	8N, 6L	2	79.5	80.3	80.5
australian-crx	690	8N, 6L	2	79.9	82.2	82.0

Table 2 - ILA - Summary Results

Space	DN'	NDL'	DHET
Nominal	85.7	82.9	85.7
Linear	78.8	83.5	83.5
Heterogeneous	80.1	83.9	84.0
Overall	81.6	83.4	84.4

As expected, *DHET=DN'* on nominal datasets, and *DHET=NDL'* on linear ones. Also, each homogeneous metric outperforms the other on its respective spaces. The above results suggest that *DHET* is an adequate metric for inductive learning in the attribute-value language. Over the selected datasets, representing a variety of input spaces, the average predictive accuracy obtained with *DHET* exceeds that obtained with either of the homogeneous metrics discussed here.

6. Conclusion

In this paper, we have addressed the problem of designing an efficient metric for heterogeneous inductive learning applications in the attribute-value language. We gave examples of the

inadequacy of homogeneous metrics, and proposed a metric, *DHET*, that effectively handles homogeneous and heterogeneous spaces. *DHET* also accounts for inductive learning algorithms that deal with rules containing don't-cares. Results of simulations with ILA were given, that confirm these findings. On average, the results with *DHET* exceed those obtained by homogeneous metrics on all spaces.

Future research includes the following extensions to *DHET*:

- Possible normalization based on standard deviations (rather than ranges)
- Non-linear normalization for linear attributes
- Individual weighting of features (likely depend upon the learning algorithm)
- Adequate treatment of *don't-know* (or missing) values.

References

[1] Aha, D.W. A Study of Instance-Based Algorithms for Supervised Learning Tasks. Technical Report, University of California, Irvine, 1991.

[2] Aha, D.W., Kibler, D., and Albert, M.K. Instance-Based Learning Algorithms. *Machine Learning*, 6, 1991, 37-66.

[3] Dasarathy, B.V. *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*. IEEE Computer Society Press, 1991.

[4] Dietterich, T.G., and Michalski, R.S. A Comparative Review of Selected Methods for Learning from Examples. In Michalski, R.S., Carbonell, J.G., and Mitchell, T.M., (Eds.). *Machine Learning: An Artificial Intelligence Approach*. Tioga Publishing Company, Palo Alto, CA, 1983, Chapter 3.

[5] Giraud-Carrier, C., and Martinez, T.R. Using Precepts to Augment Training Set Learning. In *Proceedings of the 1993 International Conference on Artificial Neural Networks and Expert Systems (ANNES'93)*, 1993, 46-51.

[6] Giraud-Carrier, C., and Martinez, T.R. ILA: Combining Inductive Learning with Prior Knowledge and Reasoning. Submitted.

[7] Kibler, D., and Aha, D.W. Learning Representative Exemplars of Concepts: An Initial Case Study. In *Proceedings of the Fourth International Workshop on Machine Learning*, 1987, 24-30.

[8] Michalski, R.S. A Theory and Methodology of Inductive Learning. *Artificial Intelligence*, 20, 1983, 111-161.

[9] Murphy, P.M., and Aha, D.W. *UCI Repository of machine learning databases*. Irvine, CA: University of California, Department of Information and Computer Science, 1992.

[10] Salzberg, S. A Nearest Hyperrectangle Learning Method. *Machine Learning*, 6, 1991, 277-309.

[11] Stanfill, C., and Waltz, D. Toward Memory-Based Reasoning. Communications of the ACM, Vol. 29, No. 12, December 1986, 1213-1228.

[12] Sun, R. A Connectionist Model for Commonsense Reasoning Incorporating Rules and Similarities. *Knowledge Acquisition*, 4, 1992, 293-321.

[13] Wettschereck, D., and Dietterich, T.G. An Experimental Comparison of the Nearest-Neighbor and Nearest-Hyperrectangle Algorithms. To appear in *Machine Learning*.