Proceedings of the International Conference on Artificial Intelligence, Expert Systems and Neural Networks (AIE'96), pp. 11-14, 1996.

Value Difference Metrics for Continuously Valued Attributes

D. Randall Wilson, Tony R. Martinez
e-mail: *randy@axon.cs.byu.edu*, *martinez@cs.byu.edu*
Computer Science Department, Brigham Young University, Provo, UT 84602, U.S.A.

Key words: instance-based learning, generalization, distance metrics, nearest neighbor

Abstract. *Nearest neighbor and instance-based learning techniques typically handle continuous and linear input values well, but often do not handle symbolic input attributes appropriately. The Value Difference Metric (VDM) was designed to find reasonable distance values between symbolic attribute values, but it largely ignores continuous attributes, using discretization to map continuous values into symbolic values. This paper presents two heterogeneous distance metrics, called the Interpolated VDM (IVDM) and Windowed VDM (WVDM), that extend the Value Difference Metric to handle continuous attributes more appropriately. In experiments on 21 data sets the new distance metrics achieves higher classification accuracy in most cases involving continuous attributes.*

1. Introduction

Nearest neighbor (NN) techniques [1][2][3], Instance-Based Learning (IBL) algorithms [4][5][6][7], and Memory-Based Reasoning methods [8] have had much success on a wide variety of applications. Such algorithms typically store some or all available training examples during learning. During generalization, a new input vector is presented to the system for classification and a distance function is used to determine how far each stored instance is from the new input vector. The stored instance or instances which are closest to the new vector are used to classify it.

There are many distance functions that can be used to decide which instance is closest to a given input vector, including Euclidean distance and Manhattan distance (defined in Section 2). These metrics work well for numerical attributes, but they do not appropriately handle symbolic (i.e., nonlinear, and perhaps unordered) attributes.

The Value Difference Metric (VDM) was introduced by Stanfill & Waltz [8] in order to provide an appropriate distance function for symbolic attributes. The Modified Value Difference Metric (MVDM) was introduced by Cost & Salzberg [9] and used in the PEBLS system [10], and modifies the weighting scheme used by the VDM. These distance metrics work well in many symbolic domains, but they do not handle continuous attributes directly. Instead, they rely upon *discretization*, which often degrades generalization accuracy [11].

This paper presents two extensions of the Value Difference Metric which allow for more appropriate use of continuous attributes. Section 2 provides more background on the original VDM and subsequent extensions to it. Section 3 introduces the Interpolated Value Difference Metric (IVDM), and Section 4 presents the Windowed Value Difference Metric (WVDM). These two sections present empirical results which show that IVDM and WVDM provide a significant improvement in classification accuracy over VDM on several data sets. Section 5 provides conclusions and future research areas.

2. Background: Value Difference Metric (VDM)

The learning systems described in this paper address the problem of *classification*. A system is presented with a *training set* T, which consists of n *instances*. Each instance has an *input vector* $\boldsymbol{x}$, and an *output class* c. The problem of classification is to decide what the output class of a new input vector $\boldsymbol{y}$ should be, based on what was learned from the training set, even if $\boldsymbol{y}$ did not appear in the training set.

As mentioned in the introduction, Nearest Neighbor, Instance-Based, and Memory-Based techniques are similar in that they store some or all of the instances in the training set, use a distance function to determine how close $\boldsymbol{y}$ is to each stored instance, and use the class of the closest instance (or closest k instances) as the predicted class of $\boldsymbol{y}$.

These techniques—and many variations within each—differ as to which instances in the training set are retained for use during generalization. More importantly for the purposes of this paper, they also differ in their method of deciding how "close" one vector is to another.

One of the most commonly used distance function is the Euclidean Distance function, which is defined as:

$$E(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{\sum_{i=1}^{m} (x_i - y_i)^2} \tag{1}$$

where $\boldsymbol{x}$ and $\boldsymbol{y}$ are two input vectors (one typically being from a stored instance, and the other an input vector to be classified) and m is the number of input variables (*attributes*) in the application. The square root is often not computed in practice, because the closest instance(s) will still be the closest, regardless of whether the square root is taken.

An alternative function, the *city-block* or *Manhattan* distance function, can require less computation and is defined as:

$$M(\boldsymbol{x}, \boldsymbol{y}) = \sum_{a=1}^{m} |x_a - y_a| \tag{2}$$

Although these distance functions are widely used, they do not appropriately handle all of the various kinds of attributes that occur in practice.

Attribute Types. An attribute can be continuous or discrete, and a discrete attribute can be linear or symbolic. A *continuous* (or *continuously-valued*) attribute uses real values, such as a person's height or a temperature reading. A *discrete* attribute is one that can have only a fixed set of values, and can be either symbolic or linear.

A *linear discrete* attribute can have only a discrete set of linear values, such as *number of children*. It can be argued that any value stored in a computer is discrete at some level. The reason continuous attributes are treated differently is that they can have so many different values that each value may appear only rarely (perhaps only once). This causes problems for algorithms (such as VDM) that depend on testing two values for equality, because two values will rarely be equal, though they may be quite close to each other.

A *symbolic* (or *nominal*) attribute is a discrete attribute whose values are not in any linear order. For example, a variable representing symptoms might have values such as *headache, sore throat, chest pains, stomach pains, ear ache,* and *blurry vision*, which could be represented by the integers 1 through 6, respectively. Using a linear distance measurement such as (1) or (2) on such values makes little sense in this case, so the above distance functions would be inappropriate for symbolic attributes.

Value Difference Metric. The Value Difference Metric (VDM) [8] was introduced to provide an appropriate distance function for symbolic attributes. The VDM defines the distance between two values x and y of an attribute a as:

$$vdm_a(x, y) = \sum_{c=1}^{C} \left| \frac{N_{a,x,c}}{N_{a,x}} - \frac{N_{a,y,c}}{N_{a,y}} \right|^q = \sum_{c=1}^{C} \left| P_{a,x,c} - P_{a,y,c} \right|^q \tag{3}$$

where

- $N_{a,x}$ is the number of instances in the training set T that had value x for attribute a;
- $N_{a,x,c}$ is the number of instances in T that had value x for attribute a and an output class c;
- C is the number of output classes in the problem domain;
- q is a constant, usually 1 or 2; and
- $P_{a,x,c}$ is the conditional probability that the output class is c given that attribute a has the value x. As can be seen from (3), $P_{a,x,c}$ is defined as:

$$P_{a,x,c} = \frac{N_{a,x,c}}{N_{a,x}} \tag{4}$$

Note that $N_{a,x}$ is the sum of $N_{a,x,c}$ over all classes, i.e.,

$$N_{a,x} = \sum_{c=1}^{C} N_{a,x,c} \tag{5}$$

Using this distance measure $vdm_a(x,y)$, two values are considered to be closer if they have more similar classifications (i.e., more similar correlations with the output classes), regardless of what order the values may be given in. In fact, linear discrete attributes can have their values remapped randomly without changing the resultant distance measurements.

The original VDM algorithm makes use of instance weights that are not included in the above equations, and some extensions of VDM have used alternate weighting schemes. Space does not allow an explanation in this paper of all of the weighting schemes. The improvements presented in this paper are independent of such schemes. Any of the various available weighting schemes can be used in conjunction with the new distance functions presented here.

Subsequent Extensions. There have been several extensions made to the original distance function presented by Stanfill & Waltz. Cost & Salzberg [9] used Equation (3) with q=1, which they found to be approximately as accurate and computationally less expensive than q=2 as used by Stanfill & Waltz. They also modified the instance weighting scheme, and implemented their algorithm in a system called PEBLS [10].

Domingos [12] presented a system which combines instance-based learning with inductive rules. His system used Equation (3) with q=1 for symbolic attributes and a normalized linear difference for linear attributes.

Wilson & Martinez [13] introduced a Radial Basis Function (RBF) neural network that used Equation (3) with q=2 for symbolic attributes and normalized Euclidean distance for linear attributes. The attribute distances were weighted in order to bring the distance for each attribute into an approximately equal range.

One problem with the formulas presented above is that they do not define what should be done when a value appears in a new input vector that never appeared in the training set. If attribute a never has value x in any instance in the training set, then $N_{a,x,c}$ for all c will be 0, and $N_{a,x}$ (which is the sum of $N_{a,x,c}$ over all classes) will thus be 0 as well. In such cases $P_{a,x,c}$ = 0/0, which is undefined. In this paper we assign $P_{a,x,c}$ the value of 0 in such cases.

This problem becomes especially evident if one attempts to use this distance function on continuous attributes. In such cases, the values can all potentially be unique, in which case $N_{a,x}$ would be 1 for every value x, and $N_{a,x,c}$ would be 1 for one value of c and 0 for all others for a given value x. In addition, new vectors are likely to have unique values, resulting in the division by zero problem above. Even if the value of 0 is substituted for 0/0, the resulting distance measurements would be nearly useless.

Even if all values are not unique, there are typically enough different values in a continuous attribute that the statistical sample will be unreliably small for each value, and the distance measures would still be untrustworthy.

Because of these problems, it is quite inappropriate to use the VDM directly on continuous attributes. One solution to this problem, as explored by Domingos [12] and Wilson & Martinez [13], is to use a *heterogeneous* distance function. A heterogeneous distance function can use a different distance metric for each attribute, based on what kind of attribute it is [14].

Another approach to this problem is *discretization* [11][15]. Some models that have used the VDM or extensions of it (notably PEBLS [9][10]) have discretized continuous attributes into a somewhat arbitrary number of discrete ranges, and then treated these values as symbolic (discrete unordered) values. This method has the advantage of generating a large enough statistical sample for each symbolic value that the P values have some significance. However, discretization can throw away much of the important information available in the continuous values, and thus can reduce generalization accuracy [11].

In this paper, we propose a new alternative, which is to use discretization in order to collect statistics and determine good values of $P_{a,x,c}$ for continuous values occurring in the training set instances, but then retain the continuous values for later use. During generalization, the value of $P_{a,y,c}$ for a value y is interpolated between two other values of P, namely, $P_{a,x1,c}$ and $P_{a,x2,c}$, where $x_1 \leq y \leq x_2$. Two methods for accomplishing this, IVDM and WVDM, are presented in the next two sections, respectively.

A generic version of the VDM algorithm, called the *discretized value difference metric* (DVDM) will be used for comparisons with extensions presented in this paper. It differs from IVDM and WVDM only in the distance function it uses. Thus, differences between DVDM and the new functions can be applied to the original VDM algorithm or other extensions such as PEBLS.

3. Interpolated Value Difference Metric (IVDM)

As explained above, the VDM uses statistics derived from the training set instances to determine a probability $P_{a,x,c}$ that the output class is c given the input value x for attribute a. Finding these probability values constitutes the *learning phase* of the Interpolated Value Difference Metric (IVDM) as well as the Discretized (i.e., default) Value Difference Metric (DVDM) algorithms.

In both DVDM and IVDM, continuous values are discretized into s equal-width intervals, where s is an integer supplied by the user. Unfortunately, there is very little guidance on what value of s to use. A value that is too large will reduce the statistical strength of the values of P, while a value too small will not allow for discrimination among classes. For the purposes of this paper, we use a simple heuristic to determine s automatically: let s be 5 or C, whichever is greatest, where C is the number of output classes in the problem domain. Current research is examining more sophisticated techniques for determining good values of s [11].

The width w_a of a discretized interval for attribute a is:

$$w_a = \frac{|max_a - min_a|}{s} \tag{6}$$

where max_a and min_a are the maximum and minimum value for attribute a, respectively, occurring in the training set.

As an example, consider the *Iris* database, available as part of the Machine Learning

Databases at the University of California Irvine (UCI) [16]. The *Iris* database has four continuous input attributes, the first of which is *sepal length*. Let T be a training set consisting of 90% of the 150 available training instances, and S be a test set consisting of the remaining 10%.

In one such division, the values for the *sepal length* attribute ranged from 4.3 to 7.9. There are only three output classes in this database, so we let s=5, resulting in a width of |7.9 - 4.3| / 5 = 0.72. Note that since the discretization is part of the learning process, it would be unfair to use any instances in the test set to help determine how to discretize the values. The discretized value v of a continuous value x for attribute a is an integer from 1 to s, and is given by:

$$v = disc_a(x) = \begin{cases} \left\lfloor \dfrac{(x - \min_a)}{w_a} \right\rfloor + 1, & \text{if } a \text{ is continuous} \\ x, & \text{if } a \text{ is discrete} \end{cases} \tag{7}$$

(unless x=max$_a$, in which case we must subsequently subtract 1 from v).

After deciding upon s and finding w_a, the discretized values of continuous attributes can be used just like discrete values of symbolic attributes in finding $P_{a,x,c}$. Figure 1 gives pseudo-code for how this is done.

```
for each attribute a
   for each instance i in T
      let x be the input value for attribute a of instance i.
      v = discretize_a(x) [which is just x if a is discrete]
      let c be the output class of instance i.
      increment N_a,v,c by 1.
      increment N_a,v by 1.
   for each value v (of attribute a)
      for each class c
         if N_a,v=0
            P_a,v,c=0
         else P_a,v,c = N_a,v,c / N_a,v
```

Figure 1. Pseudo code for finding $P_{a,x,c}$.

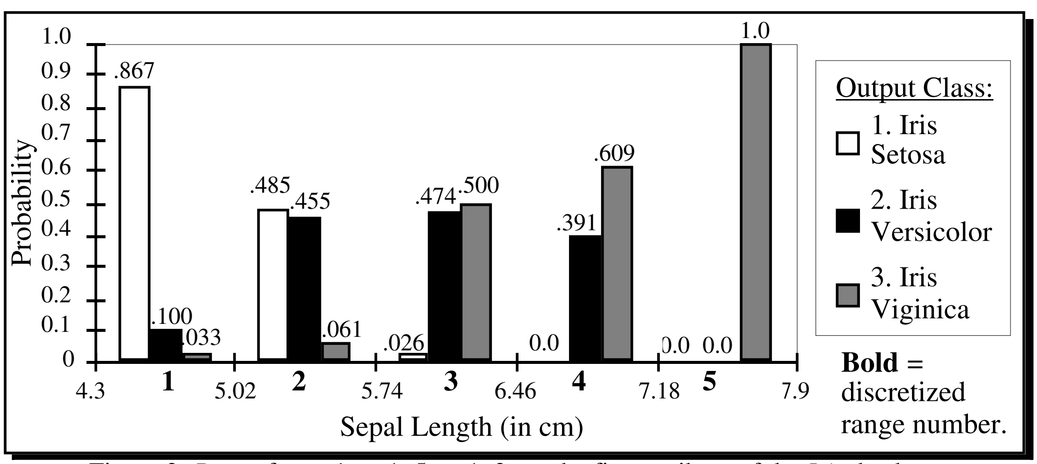

Figure 2. $P_{a,x,c}$ for a=1, x=1..5, c=1..3, on the first attribute of the *Iris* database.

For the first attribute of the *Iris* database, the values of $P_{a,x,c}$ are displayed in Figure 2. For each of the five discretized ranges of x, the probability for each of the three corresponding output classes are shown as the bar heights. Note that the heights of the three bars sum to 1.0 in each case. The bold integers indicate the discretized value of each range. For example, a sepal length greater than or equal to 5.74 but less than 6.46 would have a discretized value of “3”.

Thus far the DVDM and IVDM algorithms learn identically. However, at this point, the DVDM algorithm need not even retain the original continuous values, for it will use only the discretized values during generalization. On the other hand, the IVDM will still make use of the continuous values.

Generalization. During generalization, a new vector $\boldsymbol{y}$ is presented to the system for classification. The distance between $\boldsymbol{y}$ and the input vector $\boldsymbol{x}$ of each instance in the training set is computed, and the output class of the instance which has the minimum distance to $\boldsymbol{y}$ is used as the output class of $\boldsymbol{y}$. IVDM and DVDM differ only in how they compute the distance between $\boldsymbol{y}$ and $\boldsymbol{x}$.

DVDM uses the following distance function:

$$DVDM(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{m} \left| vdm_i(disc_i(x_i), disc_i(y_i)) \right|^2 \tag{8}$$

where $disc_i$ is athe discretization function defined in Equation (7) and vdm_i is defined as in Equation (3), with q=2. We repeat it here for convenience:

$$vdm_a(x,y) = \sum_{c=1}^{C} \left| P_{a,x,c} - P_{a,y,c} \right|^2 \tag{9}$$

Unknown input values [17] are treated as simply another discrete value, as was done in [12]. As an example, consider two training instances A and B as shown in Figure 3, and a new input vector $\boldsymbol{y}$ to be classified.

	Input Attributes					
	1	2	3	4		Output Class
A:	5.0	3.6	1.4	0.2	->	1 (Iris Setosa)
B:	5.7	2.8	4.5	1.3	->	2 (Iris Versicolor)
$\boldsymbol{y}$:	5.1	3.8	1.9	0.4		

Figure 3. Two instances and one input vector from the *Iris* database.

For attribute a=1, the discretized values for A, B, and $\boldsymbol{y}$ are 1, 2, and 2, respectively. Using values from Figure 2, the distance for attribute 1 between $\boldsymbol{y}$ and A is $|.867-.485|^2 + |.1-.455|^2 + |.033-.061|^2 = .273$, while the distance between $\boldsymbol{y}$ and B is 0, since they have the same discretized value (i.e., $|.485-.485|^2 + |.455-.455|^2 + |.061-.061|^2 = 0$).

Note that $\boldsymbol{y}$ and B have values on different ends of range 2, and are not actually nearly as close as $\boldsymbol{y}$ and A are. In spite of this fact, the discretized distance function says that $\boldsymbol{y}$ and B are equal because they fall into the same discretized range.

IVDM uses a distance function that uses interpolation to alleviate such problems. IVDM assumes that the $P_{a,x,c}$ value holds true only at the midpoint of each range, and interpolates between midpoints to find P for other attribute values.

Figure 4 shows the P values for the second output class (*Iris Versicolor*) as a function of the first attribute value (*sepal length*). The dashed line indicates what P value would be used by DVDM, and the solid line shows what IVDM would use.

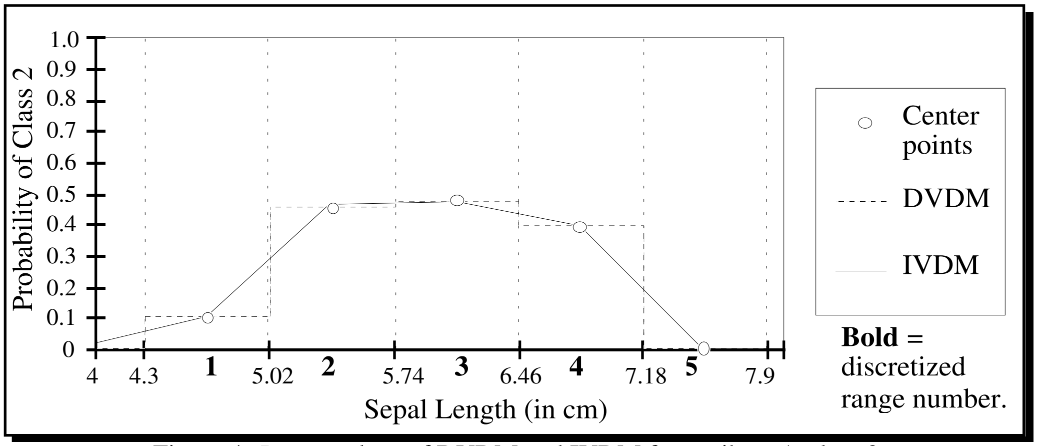

Figure 4. $P_{1,x,2}$ values of DVDM and IVDM for attribute 1, class 2.

The distance function for the Interpolated Value Difference Metric is defined as:

$$IVDM(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{m} ivdm_i(x_i, y_i)^2 \tag{10}$$

where $ivdm_i$ is defined as:

$$ivdm_a(x,y) = \begin{cases} vdm_a(x,y), & \text{if } a \text{ is discrete} \\ \sum_{c=1}^{C} \left| p_{a,c}(x) - p_{a,c}(y) \right|^2, & \text{otherwise} \end{cases} \tag{11}$$

The formula for determining the interpolated probability value $p_{a,c}(x)$ of a continuous value x for attribute a and class c is:

$$p_{a,c}(x) = P_{a,u,c} + \left(\frac{x - mid_{a,u}}{mid_{a,u+1} - mid_{a,u}} \right) * (P_{a,u+1,c} - P_{a,u,c}) \tag{12}$$

In this equation, $mid_{a,u}$ and $mid_{a,u+1}$ are midpoints of two consecutive discretized ranges such that $mid_{a,u} \leq x < mid_{a,u+1}$. $P_{a,u,c}$ is the probability value of the discretized range u, which is taken to be the probability value of the midpoint of range u (and similarly for $P_{a,u+1,c}$). The value of u is found by first setting $u = discretize_a(x)$, and then subtracting 1 from u if $x < mid_{a,u}$. The value of $mid_{a,u}$ can be found from Equation (13):

$$mid_{a,u} = min_a + width_a * (u + .5) \tag{13}$$

Figure 5 shows the values of $p_{a,c}(x)$ for attribute 1 of the *Iris* database. Since there are no data points outside the range $min_a..max_a$, the probability value $P_{a,u,c}$ is taken to be 0 when $u<1$ or $u>s$, which can be seen visually by the diagonal lines sloping toward zero on the outer edges of the graph.

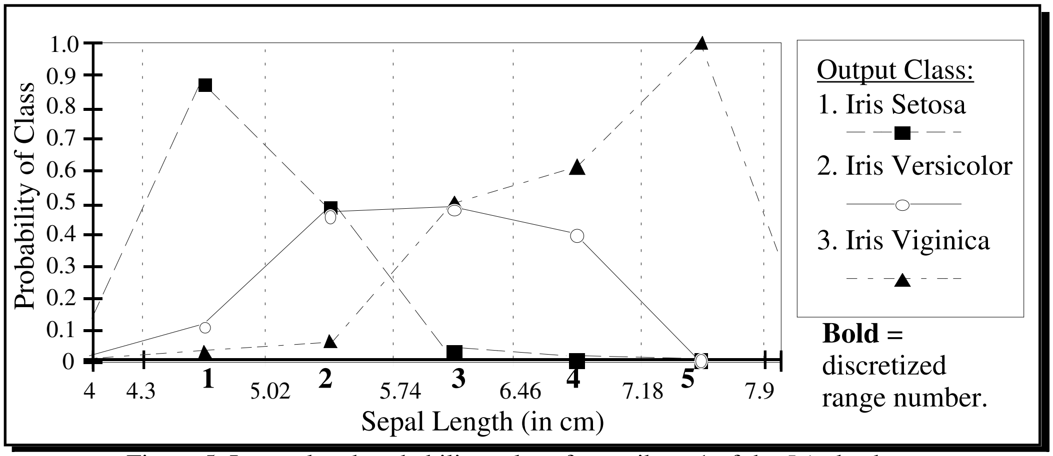

Figure 5. Interpolated probability values for attribute 1 of the *Iris* database.

Using IVDM on the example instances in Figure 3, the values for the first attribute are not discretized as they are with DVDM, but are used to find interpolated probability values. In that example, ***y*** has a value of 5.1, so $p_{1,c}(x)$ interpolates between midpoints 1 and 2, returning the values shown in Figure 6 for each of the three classes. Instance *A* has a value of 5.0, which also falls between midpoints 1 and 2, but instance *B* has a value of 5.7, which falls between midpoints 2 and 3. As can be seen from Figure 6, IVDM (using the single-attribute distance function *ivdm*) returns a distance which indicates that ***y*** is much closer to *A* than *B* (for the first attribute), which is certainly the case here. DVDM (using the discretized *vdm*), on the other hand, returns a distance which indicates that the value of ***y*** is *equal* to that of *B*, and quite far from *A*, illustrating the problems involved with using discretization.

	value	$p_{1,1}(v)$	$p_{1,2}(v)$	$p_{1,3}(v)$	$ivdm_1(v,y)$	$vdm_1(v,y)$
A	5.0	.687	.268	.046	**.005**	.273
B	5.7	.281	.463	.256	**.188**	0
y	5.1	.634	.317	.050		

Figure 6. Example of *ivdm* vs. *vdm*.

The IVDM and DVDM algorithms were implemented and tested on 21 databases from the Machine Learning Database Repository at the University of California Irvine [16]. These data sets were selected because they contain at least some continuous attributes. Recall that on domains with all symbolic attributes, IVDM and DVDM are equivalent.

Each test consisted of ten trials, each using one of ten partitions of the data randomly selected from the data sets, i.e., 10-fold cross-validation [18]. Each trial consisted of learning from 90% of the training instances, and then seeing how many of the remaining 10% of the instances were classified correctly. Both IVDM and DVDM used the same training sets and test sets for each trial.

The average accuracy for each database over all trials is shown in Figure 7. A bold value indicates which value was highest for each database. One asterisk (*) indicates that the difference is statistically significant at a 90% confidence level, using a one-tailed paired *t*-test. Two asterisks (**) are used to mark differences that are significant at a 95% or higher confidence interval.

On this set of applications, IVDM obtained higher generalization accuracy than the discretized algorithm in 15 out of 21 cases, 5 of which were significant at the 95% level or above. DVDM had a higher accuracy in six cases, but only one of those had a difference that was statistically significant.

The above results indicate that the interpolated distance function is typically more appropriate than the discretized value difference metric for applications with one or more continuous attributes. Recall that for applications with no continuous attributes, IVDM and DVDM are equivalent.

Database	DVDM	IVDM
Annealing	99.37	**99.50**
Australian Credit	**83.04***	80.58
Credit Screening	80.14	**80.43**
Flags	**58.76**	58.18
Glass	56.06	**71.02** **
Heart (Cleveland)	53.82	**54.47**
Heart (Hungarian)	79.93	**81.30**
Heart (Long-Beach-VA)	**24.50**	21.00
Heart (More)	44.65	**44.97**
Heart (Swiss)	**34.81**	30.19
Heart	80.37	**81.85**
Hepatitis	79.83	**82.54** *
Horse-Colic	**82.42**	82.11
Ionosphere	**92.60**	91.17
Iris	92.00	**94.67**
Liver Disorders	55.04	**58.51**
Satellite Image	87.06	**89.79** **
Shuttle	96.17	**99.75** **
Sonar	78.45	**84.17** *
Vowel	91.47	**97.53** **
Wine	94.38	**97.78** **
Average:	73.57	**75.31**

Figure 7. Generalization accuracy for DVDM vs. IVDM.

4. Windowed Value Difference Metric (WVDM)

The IVDM algorithm can be thought of as sampling the value of $P_{a,u,c}$ at the midpoint $mid_{a,u}$ of each discretized range u. P is sampled by first finding the instances that have a value for attribute a in the range $mid_{a,u} \pm w_a/2$. $N_{a,u}$ is incremented once for each such instance, and $N_{a,u,c}$ is also incremented for each instance whose output class is c, after which $P_{a,u,c}=N_{a,u,c}/N_{a,u}$ is computed. IVDM then interpolates between these sampled points to provide a continuous but rough approximation to the function $p_{a,c}(x)$. It is possible to sample P at more points and thus provide a better approximation to the function $p_{a,c}(x)$, which may in turn provide for more accurate distance measurements between values.

The Windowed Value Difference Metric (WVDM) samples the value of $P_{a,x,c}$ for each attribute a at each point x occurring in the training set, instead of only at the midpoints of each range. In fact, the discretized ranges are not even used by WVDM on continuous attributes, except to determine an appropriate *window width*, w_a, which is the same as the range width used in DVDM and IVDM.

For each point x occurring in the training set for attribute a, P is sampled by finding the instances that have a value for attribute a in the range $x \pm w_a/2$, and then computing $N_{a,x}$, $N_{a,x,c}$, and $P_{a,x,c}=N_{a,x,c}/N_{a,x}$ as before. Thus, instead of having a fixed number of sampling points, a *window* of instances, centered on each training instance, is used for determining the probability at a given point.

Values occurring between these sampled points are interpolated just as before, except that there are now many more points available, so a new value will be interpolated between two closer, more precise values than with IVDM.

The main drawbacks to this approach are an increased learning time, and the increased storage needed to retain C probability values for each attribute value in the training set. Execution time is not significantly increased over IVDM or DVDM.

Figure 8 shows the probability values for each of the three classes for the first attribute of the *Iris* database again, this time using the windowed sampling technique. Comparing Figure 8 with Figure 5 reveals that on this attribute IVDM provides approximately the same overall shape, but misses much of the detail.

Note, for example, the peak occurring for output class 2 at approximately *sepal length*=5.75. In Figure 5 there is a flat line which misses this peak entirely, due mostly to the somewhat arbitrary position of the midpoints at which the probability values are sampled.

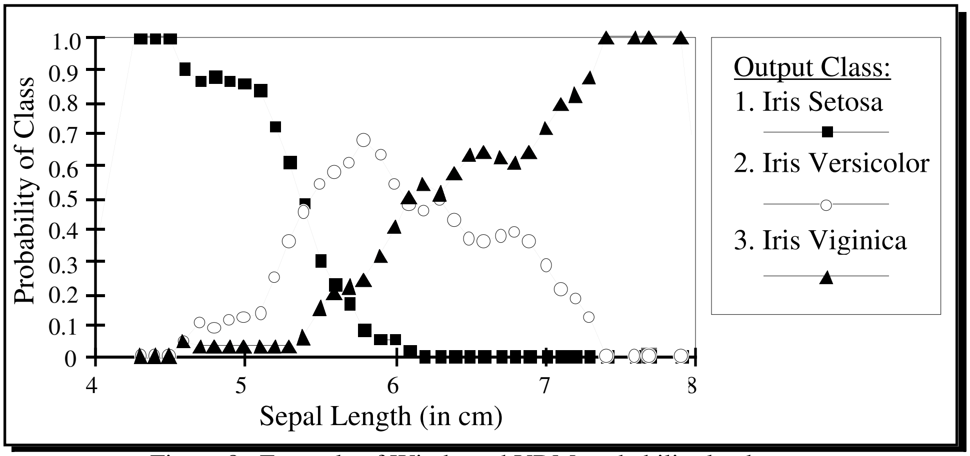

Figure 8. Example of Windowed VDM probability landscape.

The WVDM algorithm was tested on the same data sets as DVDM and IVDM, and the results are summarized in Figure 9. A bold entry indicates the highest of the three accuracy measurements, and asterisks indicate values for which the difference between WVDM and DVDM is statistically significant. The classification accuracy of IVDM is provided for convenience in comparing all three algorithms, though it is not included in the significance tests indicated in this figure.

On this set of databases, WVDM had a higher average accuracy than DVDM on 16 out of the 21 databases, and was significantly higher on 9, while DVDM was only higher on 4 databases, and none of those differences were statistically significant.

WVDM had a higher accuracy than IVDM on 12 databases, while IVDM was higher than WVDM on 8. WVDM performed slightly better than IVDM overall, and had the best average accuracy of the three algorithms.

All three of these algorithms performed quite poorly on some of the *heart* databases. Further investigation into this phenomenon may help to reveal some of the conditions that determine whether the value difference metric (and its derivatives) is an appropriate distance function for a given database.

Overall, the results of these experiments are quite encouraging. These new distance metrics can be easily incorporated into existing systems such as PEBLS in order to provide more accurate generalization in many domains that contain continuous attributes.

Database	DVDM	IVDM	WVDM
Annealing	99.37	**99.50**	99.37
Australian Credit	**83.04**	80.58	82.46
Credit Screening	80.14	80.43	**82.32** **
Flags	58.76	58.18	**59.26**
Glass	56.06	71.02	**72.47** **
Heart (Cleveland)	53.82	54.47	**57.14** *
Heart (Hungarian)	79.93	81.30	**81.61**
Heart (Long-Beach-VA)	**24.50**	21.00	22.00
Heart (More)	44.65	**44.97**	44.77
Heart (Swiss)	34.81	30.19	**35.90**
Heart	80.37	81.85	**82.59**
Hepatitis	79.83	**82.54**	78.63
Horse-Colic	82.42	82.11	**84.41**
Ionosphere	**92.60**	91.17	91.44
Iris	92.00	94.67	**96.00** *
Liver Disorders	55.04	**58.51**	56.50
Satellite Image	87.06	**89.79**	89.36 **
Shuttle	96.17	**99.75**	99.65 **
Sonar	78.45	**84.17**	**84.17** *
Vowel	91.47	**97.53**	96.21 **
Wine	94.38	**97.78**	96.67 *
Average:	73.57	75.31	**75.85**

Figure 9. Classification accuracy of WVDM vs. DVDM.

5. Conclusions & Future Research Areas

The Value Difference Metric (VDM) was designed to provide an appropriate measure of distance between two symbolic attribute values. Current systems that make use of the VDM typically discretize continuous data into discrete ranges, which causes a loss of information and often a corresponding loss in generalization accuracy.

This paper introduced two new distance functions, the Interpolated Value Difference Metric (IVDM) and Windowed Value Difference Metric (WVDM) which seek to provide a more appropriate way of handling continuous attributes within the same paradigm as VDM. Both IVDM and WVDM provide classification accuracy which is higher than the discretized version of the algorithm (DVDM) on typical databases with continuous attributes, and they are both equivalent to VDM on applications without any continuous attributes.

Current research is looking at automatically determining the width of discretization ranges or windows, and is determining whether different-sized windows might be more effective than a static window size. Also, continuing research will compare IVDM and WVDM to heterogeneous distance functions to find when each is most appropriate, and to determine under what conditions each algorithm tends to perform poorly.

Both IVDM and WVDM can be incorporated into other systems such as PEBLS to handle continuous values, and can be used in conjunction with weighting schemes and other improvements that each system provides. The results of this research are encouraging, and indicate that continuous values can be handled directly in the VDM paradigm without discretization, and that doing so improves accuracy over discretized methods.

Bibliography

[1] Cover, T. M., and P. E. Hart, (1967). "Nearest Neighbor Pattern Classification," Institute of Electrical and Electronics Engineers Transactions on Information Theory, vol. 13, no. 1, January 1967, pp. 21-27.

[2] Hart, P. E., (1968). "The Condensed Nearest Neighbor Rule," *Institute of Electrical and Electronics Engineers Transactions on Information Theory*, vol. 14, pp. 515-516.

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

[4] Aha, David W., Dennis Kibler, Marc K. Albert, (1991). "Instance-Based Learning Algorithms," *Machine Learning*, vol. 6, pp. 37-66.

[5] Aha, David W., (1992). "Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms," *International Journal of Man-Machine Studies*, vol. 36, pp. 267-287.

[6] Wilson, D. Randall, (1994). *Prototype Styles of Generalization*, Master's Thesis, Brigham Young University.

[7] Wettschereck, Dietrich, David W. Aha, and Takao Mohri, (1995). "A Review and Comparative Evaluation of Feature Weighting Methods for Lazy Learning Algorithms," Technical Report AIC-95-012, Washington, D.C.: Naval Research Laboratory, Navy Center for Applied Research in Artificial Intelligence.

[8] Stanfill, C., and D. Waltz, (1986). "Toward memory-based reasoning," *Communications of the ACM*, vol. 29, December 1986, pp. 1213-1228.

[9] Cost, Scott, and Steven Salzberg, (1993). "A Weighted Nearest Neighbor Algorithm for Learning with Symbolic Features," *Machine Learning*, vol. 10, pp. 57-78.

[10] Rachlin, John, Simon Kasif, Steven Salzberg, David W. Aha, (1994). "Towards a Better Understanding of Memory-Based and Bayesian Classifiers," in *Proceedings of the Eleventh International Machine Learning Conference,* New Brunswick, NJ: Morgan Kaufmann, pp. 242-250.

[11] Ventura, Dan, (1995). *On Discretization as a Preprocessing Step for Supervised Learning Models*, Master's Thesis, Brigham Young University, 1995.

[12] Domingos, Pedro, (1995). "Rule Induction and Instance-Based Learning: A Unified Approach," to appear in *The 1995 International Joint Conference on Artificial Intelligence (IJCAI-95)*.

[13] Wilson, D. Randall, and Tony R. Martinez, (1995). "Heterogeneous Radial Basis Functions," to appear in *Proceedings of the 1996 International Conference on Neural Networks (ICNN'96)*.

[14] Giraud-Carrier, Christophe, and Tony Martinez, (1995). "An Efficient Metric for Heterogeneous Inductive Learning Applications in the Attribute-Value Language," *Intelligent Systems*, pp. 341-350.

[15] Lebowitz, Michael, (1985). "Categorizing Numeric Information for Generalization," *Cognitive Science*, vol. 9, pp. 285-308.

[16] Murphy, P. M., and D. W. Aha, (1993). *UCI Repository of Machine Learning Databases*. Irvine, CA: University of California Irvine, Department of Information and Computer Science. Internet: ftp://ftp.ics.uci.edu/pub/machine-learning-databases.

[17] Quinlan, J. R., (1989). "Unknown Attribute Values in Induction," *Proceedings of the 6th International Workshop on Machine Learning*, San Mateo, CA: Morgan Kaufmann, pp. 164-168.

[18] Schaffer, Cullen, (1993). "Selecting a Classification Method by Cross-Validation," *Machine Learning*, vol. 13, no. 1.