JMLR: Workshop and Conference Proceedings 10: 95-104 The Fourth Workshop on Feature Selection in Data Mining

Learning Dissimilarities for Categorical Symbols

Jierui Xie XIEJ2@CS.RPI.EDU
Boleslaw Szymanski SZYMANSK@CS.RPI.EDU
Mohammed J. Zaki ZAKI@CS.RPI.EDU
Department of Computer Science
Rensselaer Polytechnic Institute
Troy, NY 12180, USA

Editor: Huan Liu, Hiroshi Motoda, Rudy Setiono, and Zheng Zhao

Abstract

In this paper we learn a *dissimilarity* measure for categorical data, for effective classification of the data points. Each categorical feature (with values taken from a finite set of symbols) is mapped onto a continuous feature whose values are real numbers. Guided by the classification error based on a nearest neighbor based technique, we repeatedly update the assignment of categorical symbols to real numbers to minimize this error. Intuitively, the algorithm pushes together points with the same class label, while enlarging the distances to points labeled differently. Our experiments show that 1) the learned dissimilarities improve classification accuracy by using the affinities of categorical symbols; 2) they outperform dissimilarities produced by previous data-driven methods; 3) our enhanced nearest neighbor classifier (called LD) based on the new space is competitive compared with classifiers such as decision trees, RBF neural networks, Naïve Bayes and support vector machines, on a range of categorical datasets.

Keywords: Dissimilarity, Categorical Data, Learning Algorithm, Classification, Feature Selection

1. Introduction

The notion of *distance* plays an important role in many data mining tasks, such as classification, clustering, and outlier detection. However, the notion of distance for categorical data is rarely defined precisely, if at all. By categorical or symbolic data, we refer to values that are nominal (e.g. colors) or ordinal (e.g. rating, typically imposed subjectively by a human). In many cases, the *dissimilarities* between symbols are fuzzy and often arbitrary. An example could be the rating of a movie, chosen from the list "very bad, bad, fair, good, very good". It is hard to determine how much one symbol differs from another. In this paper, we introduce a new method to derive dissimilarities between categorical symbols in such a way that the power of distance-based data mining methods can be applied.

The notations used throughout the paper are as follows. There is a dataset $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t\}$ of t data points, where each point $\mathbf{x}_i$ is a tuple of m attributes values, $\mathbf{x}_i = (x_i^1, \ldots, x_i^m)$. Each of the m attributes A^i is categorical, i.e., the attribute values for A^i are drawn from a set of n_i discrete values given as $\{a_1^i, a_2^i, \ldots, a_{n_i}^i\}$, which also constitute the domain of A^i. We assume that all symbols across all attributes are unique. For simplicity, we use the notation A^i to refer to the i-th attribute, as well as the domain of that attribute. Each a^i_j is also called a *symbol*. Each data point $\mathbf{x}_i$ (in the training set) also has associated with the "true" class label, given as $L(\mathbf{x}_i)$. In this paper we only consider the case where there are two classes, i.e., $L(\mathbf{x}_i) \in \{1, 2\}$, where 1 and 2 are the two class labels.

The *similarity* between symbols a^i_k and a^i_l of an attribute A^i is denoted as $S(a^i_k, a^i_l)$, whereas the *dissimilarity* or *distance* between two symbols is denoted as $D(a^i_k, a^i_l)$. Typically $S(a^i_k, a^i_l) : A^i \times A^i \rightarrow (0, 1)$, in which case $D(a^i_k, a^i_l) = 1 - S(a^i_k, a^i_l)$. In other cases $S(a^i_k, a^i_l) : A^i \times A^i \rightarrow \mathbb{R}^+$, in which case $D(a^i_k, a^i_l) = \frac{1}{S(a^i_k, a^i_l)}$. The distance between two data points $\mathbf{x}_i$ and $\mathbf{x}_j$ is defined in terms of the distance between symbols, as follows:

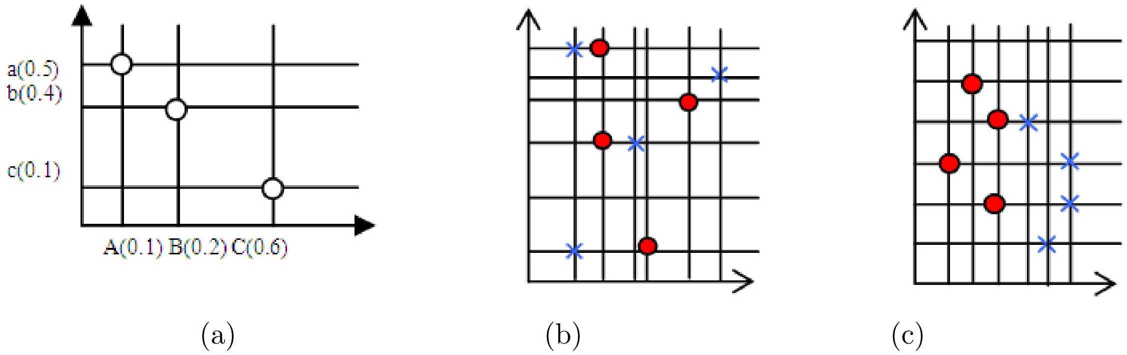
$$D(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\sum_{k=1}^{m} D(x^k_i, x^k_j)^2} \tag{1}$$

Given a point $\mathbf{x}_i$, the error of a classifier on that point is defined as:

$$e_{\mathbf{x}_i} = \frac{(L(\mathbf{x}_i) - O(\mathbf{x}_i))^2}{2} \tag{2}$$

where $O(\mathbf{x}_i)$ is the output class of the classifier on point $\mathbf{x}_i$. Since $O(\mathbf{x}_i) \in \{1, 2\}$, $e_{\mathbf{x}_i} \in \{0, \frac{1}{2}\}$. The total error rate of the classifier on a set of t points is simply $E = \sum_{i=1}^{t} e_{\mathbf{x}_i}$.

In this paper, our goal is to learn a mapping function from each categorical attribute A^i onto the real number interval, given by the function $r : A^i \rightarrow \mathbb{R}$, with the aim of minimizing the total error rate E. Once r has been learned, each categorical data point $\mathbf{x}_i$ can be treated as a m-dimensional point or vector in $\mathbb{R}^m$, given as $r(\mathbf{x}_i) = (r(x_{i,1}), \ldots, r(x_{i,m}))^T$. This enables one to apply any of the distance-based classification methods directly on the transformed dataset $r(X) = \{r(\mathbf{x}_i)\}_{i=1}^{t}$.



(a) (b) (c)

Figure 1: (a) Mapping symbols to real values. Dataset consists of three points: $\mathbf{x}_1 = (A, a)$, $\mathbf{x}_2 = (B, b)$ and $\mathbf{x}_3 = (C, c)$. The mapping function r is given as: $r(A) = 0.1$, $r(B) = 0.2$, $r(C) = 0.6$, $r(a) = 0.5$, $r(b) = 0.4$ and $r(c) = 0.1$. (b) Random mapping. (c) Linearly separable mapping.

As an example, consider Figure 1 (a), which shows three points over two categorical attributes "color" (with symbols A, B and C) and "shape" (with symbols a, b and c). In the new continuous space, a value assignment to a symbol naturally defines a hyperplane that contains all the points in the dataset having that particular symbol. In this example, each point is defined by exactly two straight lines. Figure 1 (b) shows an extended example with 8 points, with a random initial mapping, which does not discriminate too well between the two classes. Our goal is to improve the initial mapping into a classification-aware mapping like that in Figure 1 (c), which achieves a low classification error rate (on the training set, and hopefully on the test set too).

2. Related Work

The most widely used measure on categorical data is simple matching (or overlap), which is defined as $S(a_i, a_j) = 1$ if $a_i = a_j$ and $S(a_i, a_j) = 0$, otherwise. This measure simply checks that two symbols are the same, which forms the basis for various distance functions, such as Hamming and Jaccard distance (Liang, 2004).

The simple matching ignores information from the dataset and the desired classification. Therefore, many more data-driven measures have been developed to capture preferences for matching or mismatching based on symbols' statistics. Here, we divide related methods into two categories: *unsupervised* and *supervised* methods. The unsupervised methods are typically based on frequency or entropy. Let $f(a_i)$ be the frequency of symbol a_i of attribute A in the dataset, then $p(a_i) = f(a_i)/t$.

Let a_i and a_j be two symbols in the domain of attribute A. Lin (1998) defines $S(a_i, a_j) = 2\log p(a_i)$ if $a_i = a_j$, and $2\log(p(a_i)+p(a_j))$, otherwise, which gives more weight to matches on frequent values and lower weight to mismatches on infrequent values. Burnaby (1970) defines $S(a_i, a_j) = 1$ if $a_i = a_j$. However, if $a_i \neq a_j$, then

$$S(a_i, a_j) = \frac{\sum_{a_k \in A} 2\log(1 - p(a_k))}{\log\left(\frac{p(a_i)p(a_j)}{(1-p(a_i))(1-p(a_j))}\right) + \sum_{a_k \in A} 2\log(1 - p(a_k))}$$

Smirnov (1968) not only considers the frequency, but also takes the distribution of the other attributes values into account, defining

$$S(a_i, a_j) = 2 + \frac{t - f(a_i)}{f(a_i)} + \sum_{a_k \in A \setminus \{a_i\}} \frac{f(a_k)}{t - f(a_k)}$$

if $a_i = a_j$, and

$$S(a_i, a_j) = \sum_{a_k \in A \setminus \{a_i, a_j\}} \frac{f(a_k)}{n - f(a_k)}$$

otherwise. Goodall (1966) proposed another statistical approach, in which less frequent attribute values make greater contribution to the overall similarity than frequent attribute values. A modified version called Goodall1 is proposed in (Boriah et al., 2008), defining

$$S(a_i, a_j) = 1 - \sum_{a_k \in A, p(a_k) < p(a_i)} p^2(a_k)$$

if $a_i = a_j$, and 0 otherwise. Gambaryan (1964) proposed a measure related to information entropy, which gives more weight to matches where the number of matches is between frequent and rare. If $a_i = a_j$, the similarity is given as

$$S(a_i, a_j) = -[p(a_i)\log_2 p(a_i) + (1 - p(a_i))\log_2(1 - p(a_i))]$$

and 0 otherwise. Eskin et al. (2002) consider the number of symbols of each attribute. In its modified version (Boriah et al., 2008), this measure gives more weight to mismatches that occur on an attribute with more symbols using the weight $n^2/(n^2+2)$, where n is the number of symbols of attribute A. Occurrence Frequency (OF) (Jones, 1988) gives lower similarity to mismatches on less frequent symbols and higher similarity on mismatches on more frequent symbols. Conversely, Inverse Occurrence Frequency (IOF) assigns higher similarity to mismatches on less frequent symbols. That is, if $a_i \neq a_j$, then

$$S(a_i, a_j) = \frac{1}{1 + \log(\frac{n}{f(a_i)})\log(\frac{n}{f(a_j)})}$$

for OF, and

$$S(a_i, a_j) = \frac{1}{1 + \log(f(a_i))\log(f(a_j))}$$

for IOF. When $a_i = a_j$, both define $S(a_i, a_j) = 1$. More discussion on these kinds of measures is given by Boriah et al. (2008).

The supervised methods take advantage of the class information. An example is Value Difference Metric (VDM) proposed in (Stanfill and Waltz, 1986). The main idea is that symbols are similar if they occur with a similar relative frequency for all the classes. The dissimilarity between a_i and a_j is defined as a sum over n classes:

$$D(a_i, a_j) = \sum_{c=1}^{n} \left| \frac{C_{a_i,c}}{C_{a_i}} - \frac{C_{a_j,c}}{C_{a_j}} \right|^h$$

where $C_{a_i,c}$ is the number of times symbol a_i occurs in class c. C_{a_i} is the total number of times a_i occurs in the whole dataset. Constant h is usually set to 1. Cheng et al. (2004) proposed an approach based on Hadamard product and RBF classifier. They attempt to evaluate all the pair-wise distances between symbols, and they optimize the error function using gradient descent method. In our algorithm the number of values to be estimated is equal to the number of symbols across all attributes, i.e. linear in the symbol set size, which may enable faster and more robust learning. However, we were unable to compare the methods directly since we did not have access to the code from Cheng et al. (2004). Furthermore, in our approach, after learning, all pair-wise distances can be easily derived if needed.

3. Learning Algorithm

Our learning algorithm is based on the gradient descent method. Starting from an initial assignment of real values to the symbols, guided by the error rate based on a nearest neighbor classifier, our method iteratively updates the assignments. Intuitively, in each iteration, the method moves the symbols (hence the lines or, more generally, the hyperplanes, as seen in Figure 1) to new locations according to the *net force* imposed on them. Let $\mathbf{x}$ be the closest point to $\mathbf{p}$ from class 1, and $\mathbf{y}$ the closest point from class 2. Let $d_1 = D(\mathbf{p}, \mathbf{x})$, and $d_2 = D(\mathbf{p}, \mathbf{y})$ be the corresponding distances, and $\Delta d = d_1 - d_2$ be the difference of the distances. Our simple nearest neighbor classifier assigns the class as follows:

$$O(\mathbf{p}) = S(\Delta d) + 1 = S(d_1 - d_2) + 1 \tag{3}$$

where $S(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function. It is easy to verify that if $d_1 \ll d_2$, then $O(\mathbf{p}) \approx 1$ and if $d_1 \gg d_2$ then $O(\mathbf{p}) \approx 2$. The classification error for $\mathbf{p}$ is $e_{\mathbf{p}}$ as given in equation (2). We update the assignment of values to symbols depending on the error $e_{\mathbf{p}}$, as discussed below. Our method in fact cycles through all points, considering each as the target, in each iteration. In batch training, the total assignment is accumulated over all the points, but in online training, the assignment is updated immediately after each point. The pseudo code of the algorithm (with batch training) is given below.

```
LD: Learning Algorithm (Dataset $X$):
  $t$ = Number of instances in the dataset;
  $r$ = Random initial asssignment;
  while(stop criteria not satisfied){
    sum$\Delta r = 0$;
    for k = 1:t{
      $\mathbf{p} = \mathbf{x}_k$ or any point taken at random from $X$;
      $d_1 = \min_{\mathbf{x}_k \in X, L(\mathbf{x}_k)=1, \mathbf{x}_k \neq \mathbf{p}}\{D(\mathbf{p}, \mathbf{x}_k)\}$;
      $d_2 = \min_{\mathbf{x}_k \in X, L(\mathbf{x}_k)=2, \mathbf{x}_k \neq \mathbf{p}}\{D(\mathbf{p}, \mathbf{x}_k)\}$;
      $\Delta d = d_1 - d_2$;
      Compute $\Delta r$ using equation (8);
      sum$\Delta r$ = sum$\Delta r + \Delta r$;
    }
    update r= r+sum$\Delta r$;
  }
```

3.1 Objective Function and Update Equation

The general update equation for a target point $\mathbf{p}$ is given as $r = r + \Delta r$. Each element in r, r_j^i, represents the real value assignment for the j-th symbol of the i-th attribute. Thus, for each r_j^i, the update equation is $r_j^i = r_j^i + \Delta r_j^i$. r_j^i moves in the direction of the negative gradient of $e_{\mathbf{p}}$ to decrease the error. That is,

$$\Delta r_j^i = -\eta \cdot \frac{\partial e_{\mathbf{p}}}{\partial a_j^i} = -\eta \cdot \frac{\partial (L(\mathbf{p}) - O(\mathbf{p}))^2/2}{\partial a_j^i} = \eta \cdot (L(\mathbf{p}) - O(\mathbf{p})) \cdot \frac{\partial O(\mathbf{p})}{\partial a_j^i} \tag{4}$$

where η is the learning rate, and the differential is taken with respect to the j-th symbol for attribute A^i, a_j^i.

Note that, by equation (3),

$$\frac{\partial O(\mathbf{p})}{\partial a_j^i} = \frac{\partial [S(d_1 - d_2) + 1]}{\partial a_j^i} = \frac{\partial S(\Delta d)}{\partial \Delta d} \cdot \frac{\partial \Delta d}{\partial a_j^i} = S(\Delta d) \cdot (1 - S(\Delta d)) \cdot \frac{\partial \Delta d}{\partial a_j^i} \tag{5}$$

The last step follows from the fact that the partial derivative of the sigmoid function $S(x)$ is given as: $\frac{\partial S(x)}{\partial x} = S(x)(1 - S(x))$.

3.2 Computing the Derivative of Δd

Note that $\frac{\partial \Delta d}{\partial a^i_j} = \frac{\partial d_1}{\partial a^i_j} - \frac{\partial d_2}{\partial a^i_j}$, where $d_1 = \sqrt{\sum_{i=1}^m D(p_i, x_i)^2}$ and $d_2 = \sqrt{\sum_{i=1}^m D(p_i, y_i)^2}$ are the distances from $\mathbf{p}$ to the closest points $\mathbf{x}$ in class 1, and $\mathbf{y}$ in class 2, respectively. Since the derivative is with respect to the j-th attribute in attribute i, even in the distance terms d_1 and d_2, only the i-th attribute has to be considered. Let us consider d_1, we have:

$$\frac{\partial d_1}{\partial a^i_j} = \frac{\partial \left(\sum_i^m D(p_i, x_i)^2\right)^{1/2}}{\partial a^i_j} = \frac{1}{2} \cdot (d_1)^{-1/2} \cdot \frac{\partial D(p_i, x_i)^2}{\partial a^i_j} \tag{6}$$

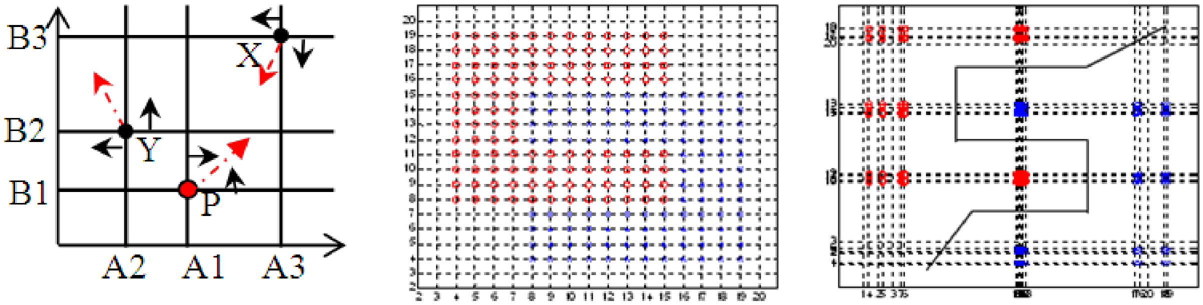
The derivative will be zero if the symbol for the i-th attribute is not a^i_j, as per the following:

$$\frac{\partial D(p_i, x_i)^2}{\partial a^i_j} = 2 \cdot D(p_i, x_i) \cdot \begin{cases} +1, & \text{if } p_i = a^i_j \text{ and } x_i \neq a^i_j \\ -1, & \text{if } p_i \neq a^i_j \text{ and } x_i = a^i_j \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

In a similar manner we can derive $\frac{\partial d_2}{\partial a^i_j}$.

By putting the above equations together we have a full version of equation (4) as follows:

$$\Delta r^i_j = \eta \cdot (L(\mathbf{p}) - O(\mathbf{p})) \cdot S(d_1 - d_2) \cdot (1 - S(d_1 - d_2)) \cdot \left(\frac{\partial d_1}{\partial a^i_j} - \frac{\partial d_2}{\partial a^i_j} \right) \tag{8}$$



(a) (b) (c)

Figure 2: (a) Schematic of forces acting on points $\mathbf{p}$, $\mathbf{x}$ and $\mathbf{y}$. Forces along each axis are in solid arrow, and net force in dashed arrow. During the learning process, $\mathbf{p}$ is getting closer to $\mathbf{x}$ and farther from $\mathbf{y}$. (b) Example 2-d synthetic data, with two features, each of which has twenty symbols. Red circles and blue stars indicate points from two classes. (c) Subspaces corresponding to the learned mapping.

3.3 Example: Line Moving and Subspace Forming

Given a target point $\mathbf{p}$ and the corresponding closest points $\mathbf{x}$ and $\mathbf{y}$, in class 1 and 2, the amount of assignment change of each symbol is proportional to Δr^i_j, which moves the location of a symbol to the left or to the right on the i-th axis (if $\Delta r^i_j < 0$ or $\Delta r^i_j > 0$

respectively). In a 2-dimensional space, the change of symbol assignments is equivalent to moving the lines around, which in turn "move" the data points to new locations.

Figure 2 (a) illustrates a case where $\mathbf{p} = (A1, B1)$ and $\mathbf{x} = (A3, B3)$ belong to class 1 but $\mathbf{p}$ is misclassified, since it is closer to $\mathbf{y} = (A2, B2)$ in class 2 (i.e., $d_1 > d_2$). In this specific case, there are six symbols, $A1, A2, A3, B1, B2$, and $B3$. Intuitively, when the learning goes on, more and more points nearby tend to get together and form a subspace containing points with the same class label. Subspaces with different class labels tend to be apart. To demonstrate how the subspaces are created, we applied our learning algorithm on an example 2-d synthetic datasets in Figure 2 (b). The learned subspace are shown in (c).

4. Discovering Symbol Redundancy

By modeling each symbol as a variable in the real space, our algorithm explores the distances between symbols on each individual feature. Interestingly, our algorithm is able to discover *redundancies* among symbols, which provides insights for improving the classification performance.

We ran our learning approach on the *Balance Scale* dataset from UCI repository (see Table 3). Table 1 shows a typical assignment learned on this dataset. As highlighted, some symbols have very close values (e.g., symbols '2'=1.76 and '3'=1.96 for attribute *left-weight*; symbols '3'=3.75 and '4'=3.47 for attribute *left-distance*, and so on). Such closeness implies that for an attribute like *left-weight* having five symbols may not be necessary for classification. We regard this kind of closeness as redundant information, which should be removed to improve the classification accuracy. To verify the above hypothesis, we merged the two closest symbols into one. For example, we replaced symbols '2' and '3' with only one symbol '23'. As shown in Table 2, the classification accuracy is improved with the merged attributes for decision tree, RBF neural network and nearest neighbor classifier (i.e., NN, based on Overlap dissimilarity measure). The merging can be considered as a form of pre-pruning process, which improves the generality of classifiers.

Table 1: A typical assignment learned on *Balance Scale* dataset

Attribute\symbol	'1'	'2'	'3'	'4'	'5'
left-weight	-0.85	**1.76**	**1.96**	4.98	7.14
left-distance	-0.72	1.78	**3.75**	**3.47**	6.71
right-weight	-0.78	2.22	3.19	**5.08**	**5.28**
right-distance	-1.04	1.10	3.68	**5.49**	**5.76**

Table 2: Accuracy improvement on merged dataset

	Original Dataset	Merged Dataset
C4.5	73.11	**76.35**
RBFNN	90.68	**92.20**
NN	75.31	**85.93**

5. Experimental Results

To evaluate the learned dissimilarity measure (short for *LD*), we compare our approach against other data-driven methods discussed in Section 2 and other popular classifiers. We present results on categorical datasets shown in Table 3, that are all taken from the UCI machine learning repository. The number of attributes ranges from 4 to 60, and each attribute takes on 2 to 12 symbols.

Table 3: Dataset Information

Dataset	Size	Dimension	Attributes and Symbols
Splice	1330	60	Each dimension takes on {A,C,T,G}
Balance Scale	576	4	Each dimension takes on {1, 2, 3, 4, 5}
Car Evaluation	768	6	buying: {v-high, high, med, low}; maint: {v-high, high, med, low}; doors: {2, 3, 4, 5+}; persons: {2, 4, more}; lug boot: {small, med, big}; safety: {low, med, high}
Connect-4	1000	42	Each dimension takes on {x, o, b}
Mushroom	1000	22	Various sizes from 2 to 12, e.g. the first attributes cap-shape: {bell, conical, convex, flat, knobbed, sunken}
Tic-tac-toe	624	9	Each dimension takes on {x, o, b}
Hayes-Roth	100	4	hobby: {1,2,3}; age: {1,2,3}; educational level: {1,2,3}; marital status: {1,2,3}

5.1 Comparison with Various Data-Driven Methods

To compare our Learned Dissimilarity approach, with those learned from other ten methods mentions in Section 2, we evaluate the classification accuracy of the nearest neighbor classifier, where the distances are computed from various dissimilarity measures. More specifically, the distance between two categorical points is calculated according to equation (1). We used 5-fold cross-validation to measure the classification accuracy. The numbers reported in Table 4, correspond to the *average classification accuracy* and standard deviation (in parenthesis) over ten runs (i.e., we repeat NN ten times for each dissimilarity measure on each dataset). The last row of Table 4 shows the average performance over all the datasets. The highest accuracy is shown in bold for each dataset.

On average, the *LD* and VDM achieve the best accuracy, indicating that supervised dissimilarities attain better results over the unsupervised counterparts. Among the unsupervised measures, IOF, Lin are slightly superior to others. Goodall1, Smirnov and OF achieve same performance as Overlap. By considering the confidence interval (accuracy +/- standard deviation) to compare the performance of different methods on each dataset, we conclude that LD performed statistically worse than Lin on datasets Splice and Tic-tac-toe but better than Lin on datasets Connection-4, Hayes and Balance Scale. Moreover, LD performed statistically worse than VDM only on one dataset (Splice) but better on two datasets (Connection-4 and Tic-tac-toe). Finally, LD performed statistically at least as well as (and on some datasets, e.g. Connection-4, better than) the remaining methods.

Table 4: Performance comparison on various dissimilarities

	Overlap	Lin	Smirnov	Goodall1	Eskin
Splice	89.45(0.58)	94.21(0.42)	88.53(0.82)	88.79(0.69)	88.42(0.60)
Balance Scale	75.31(1.44)	75.52(3.31)	74.65(2.14)	75.69(2.04)	64.23(0.51)
Car Evaluation	87.86(1.23)	92.64(1.65)	83.72(2.14)	84.96(2.26)	86.65(0.75)
Connect-4	84.20(0.92)	78.25(1.05)	75.30(0.73)	82.50(0.46)	83.55(0.30)
Mushroom	100(0)	100(0)	99.75(0.17)	99.90(0.06)	100(0)
Tic-tac-toe	81.59(1.56)	**98.64**(0.69)	84.03(1.07)	86.97(1.58)	63.85(0.93)
Hayes-Roth	70.90(4.99)	71.00(3.65)	71.00(5.50)	69.50(5.01)	67.00(5.76)
Avgerage	84.18(1.53)	87.18(1.53)	82.42(1.79)	84.04(1.72)	79.10(1.26)

	IOF	OF	Gambaryan	Burnaby	VDM	LD
Splice	90.15(0.62)	88.34(0.68)	88.38(0.67)	83.72(1.19)	**95.60**(0.57)	93.00(0.67)
Balance Scale	75.86(3.53)	75.34(3.08)	75.17(2.37)	75.43(2.88)	92.10(1.36)	**94.04**(1.21)
Car Evaluation	89.12(1.92)	92.83(0.88)	83.52(2.06)	92.44(0.86)	97.33(1.74)	**98.00**(1.47)
Connect-4	83.40(0.87)	84.70(0.78)	50.00(0)	85.15(0.84)	83.80(0.96)	**87.48**(0.92)
Mushroom	99.95(0.03)	100(0)	50.00(0)	100(0.09)	100(0)	100(0)
Tic-tac-toe	97.13(0.43)	77.48(1.17)	88.85(1.25)	69.27(1.12)	82.15(2.57)	95.30(1.72)
Hayes-Roth	68.50(4.67)	58.00(8.25)	75.50(3.52)	57.50(8.30)	73.00(4.70)	**79.40**(1.71)
Avgerage	86.30(1.72)	82.38(2.12)	73.06(1.41)	80.50(2.18)	89.14(1.70)	**92.46**(1.10)

5.2 Comparison with Various Classifiers

We consider the NN based on our learned dissimilarity as an “enhanced” nearest neighbor classifier, again denoted as *LD*. The performance of LD is compared with algorithms implemented in Weka 3.6, including decision tree (C4.5 with pruning), Naïve Bayes (NB), RBF neural network (RBFNN, with clustering technique to estimate the number of kernels), and SVM (with RBF kernel and complexity 1.0). Our method uses the learned mapping r, whereas the other methods use the Euclidean distance (corresponding to simple matching) between categorical points. The performance metric is the average classification accuracy over ten runs based on 5-fold cross validation. As shown in Table 5, considering the same confidence intervals as in Sec.5.1, we conclude that LD performed statistically worse than the other methods on only one dataset (Splice) but performed better on at least three other datasets than each of the other methods, which we believe shows a significant improvement over them.

6. Conclusions

In this paper, we propose a task-oriented or supervised iterative learning approach to learn a distance function for categorical data. The algorithm explores the relationships between categorical symbols by utilizing the classification error as guidance. We show that the real value mappings found by our algorithm provide discriminative information, which can be used to refine features and improve classification accuracy. In the future work, we would like to extend the approach to continuous and mixed attribute datasets, as well as “relational” datasets where there are links between data points.

Table 5: Performance comparison on various classifiers

	C4.5	NB	RBFNN	SVM	LD
Splice	95.03(1.00)	**97.01**(0.42)	**97.01**(0.64)	96.91(0.60)	93.00(0.67)
Balance Scale	73.11(1.78)	**96.34**(1.53)	90.68(1.50)	95.49(1.78)	94.04(1.21)
Car Evaluation	96.51(1.12)	92.32(2.33)	93.58(2.01)	88.24(1.88)	**98.00**(1.47)
Connect-4	87.01(1.71)	87.53(1.10)	**88.35**(1.62)	87.48(0.89)	87.48(0.92)
Mushroom	100(0.00)	97.33(1.00)	100(0.04)	100(0.00)	100(0.00)
Tic-tac-toe	85.44(3.26)	76.67(1.94)	80.75(2.53)	77.21(1.27)	**95.30**(1.72)
Hayes-Roth	71.00(8.07)	68.50(9.15)	72.40(5.17)	64.10(12.42)	**79.40**(1.71)
Average	86.87(2.42)	87.96(2.49)	88.97(1.93)	87.06(2.69)	**92.46**(1.10)

References

S. Boriah, V. Chandola, and V. Kumar. Similarity measures for categorical data: A comparative evaluation. In *SIAM Data Mining Conference*, pages 243–254, 2008.

T. Burnaby. On a method for character weighting a similarity coefficient employing the concept of information. *Mathematical Geology*, 2(1):25–38, 1970.

V. Cheng, C.H. Li, and J.T. Kwok. Dissimilarity learning for nominal data. *Pattern Recognition*, 37(7):1471–1477, 2004.

E. Eskin, A. Arnold, and M. Prerau. A geometric framework for unsupervised anomaly detection. *Applications of Data Mining in Computer Security*, pages 78–100, 2002.

P. Gambaryan. A mathematical model of taxonomy. *SSR*, 17(12):47–53, 1964.

D. Goodall. A new similarity index based on probability. *Biometrics*, 22(4):882–907, 1966.

K.S. Jones. A statistical interpretation of term specificity and its application in retrieval. *Document Retrieval Systems*, 3:132–142, 1988.

M. Liang. Data mining:concepts, models, methods, and algorithms. *IIE Transactions*, 36 (5):495–496, 2004.

D. Lin. An information-theoretic definition of similarity. In *15th International Conference on Machine Learning*, pages 296–304, 1998.

E. S. Smirnov. On exact methods in systematics. *Systematic Zoology*, 17(1):1–13, 1968.

C. Stanfill and D. Waltz. Toward memory-based reasoning. *CACM*, 29:1213–1228, 1986.