Society for Science and Education United Kingdom
TMLAI Transactions on Machine Learning and Artificial Intelligence
Volume 2, Issue 5
ISSN: 2054 - 7390

Classifying an Object using Class Differentiators

[1]Seunghyun Im, [2]Li-Shiang Tsay
[1]*Computer Science Department, University of Pittsburgh at Johnstown, USA;*
[2] *Department of Computer Systems Technology, North Carolina A&T University, USA;*
sim@pitt.edu; ltsay@ncat.edu

ABSTRACT

This paper discusses a supervised classification method. The method classifies an object using class differentiators. The class differentiators are the smallest set of values in a class that effectively distinguish one class from the others. The class membership is determined by the degree of homology between the test object and the class differentiators. Unlike many rule based classifiers, the algorithm proposed in this paper does not require input parameters and always produces the same results from the same data set. The algorithm is designed to work with categorical data, and is particularly useful when the quantification of the data is infeasible. We present an experimental result to show the validity of the algorithm.

Keywords: Classification algorithm, Supervised Classifier, Categorical Data, Reduct, Rough Set.

1 Introduction

We present a supervised classification method for categorical data. The classification problem in this paper focuses on the prediction of the class of a test object. We assume that the training data set has condition and decision attributes. The class labels of the training data set are determined by the discrete values in the decision attribute. The class label of the test object is uncertain. We use the training data to predict the unknown class label of the test object. The prediction can be performed by measuring the similarity (or distance) between objects. The data type has great influence on the similarity measure. For categorical data, the method can be divided into two types (1) discrete similarity measure and (2) approximated similarity measure. In general, the similarity between two attribute values is either 1 or 0 in discrete similarity measure; 1 for the same symbols and 0 for the different symbols between two objects. Therefore, the amount and frequency of the overlap between the training data and the newly observed object determine how much the object is close to a class. The classifiers based on the classical Naive Bayes [3] or Classification Rule [4] fall into this category. On the other hand, the methods derived from the approximated similarity measure convert the categorical data to numeric data in order to measure the degree of similarities between non-identical symbols. (e.g. the distance between red and blue is 0.7). Many algorithms have been proposed to address the conversion problem [5]. Although they are fundamentally useful, the quantification of categorical data is still challenging due to the fact that distances are often too arbitrary or it requires an intensive phase of data preprocessing, such as discretization of data [6] or construction of ontology [7][8]. The quantification is even more difficult when data are sanitized for privacy and regulatory requirements [9][10] because we do not know the meaning of the symbols. This is the case with many dataset from medical area. The algorithm presented in this

DOI: 10.14738/tmlai.25.594
Publication Date: 17th October, 2014
URL: http://dx.doi.org/10.14738/tmlai.25.594

paper falls into the discrete similarity measure category. We developed this algorithm for the classification task that quantification is not possible.

The rule based classifier is one of the most widely studied supervised classifier for categorical data. A number of algorithms have been proposed [11]. These algorithms generate if-then style rules, and use the antecedent to measure the similarity and the consequent to predict the class label. Despite being very useful in many applications, these methods require quality threshold values, e.g. minimum support and confidence value for rule extraction. The determination of the optimal threshold value requires a deep understanding of the data and often need help from the domain expert. Another potential problem is that the result of the classification varies depending on the input parameters because the changes in these parameters result in different set of rules. The proposed algorithm does not require input parameters. This is an advantage for the classification task that requires consistent result or for non-technical users who are not familiar with the notion of the threshold value. Naive Bayes classifier is another well-known classification algorithm. It is simple and easy to use. We believe our method is as simple as the classical Naive Bayes classifier, yet it does not need any type of pseudo-count to solve the zero frequency problem.

The proposed algorithm is inspired by the concept of object reduct [2]. An object reduct is the minimal set of values of an object which differentiates the object from other objects. We extended the concept to detect patterns characterizing each class, and use them to find the most promising class label for a newly observed object. We present our method in Chapter 2, an experimental result in Chapter 3, and the conclusion in Chapter 4.

Table 1: Example of Information System.

	B	C	E	D
x_1	b_1	c_1	e_1	d_1
x_2	b_2	c_1	e_2	d_2
x_3	b_2	c_3		d_2
x_4	b_1	c_1	e_1	d_2

2 Algorithm

2.1 Basic Notations

We will use the following notations for dataset. By an information system [1] we mean a triple $S = (X,A,V)$, where

$X = \{x_1, x_2, ..., x_i\}$ *is a finite set of objects,*

$A = \{a_1, a_2, ..., a_j\}$ *is a finite set of attributes*

$V = \{v_1, v_2, ..., v_k\}$ *is a finite set of attribute values.*

The information system in Table 1 has 4 objects $X = \{x_1, x_2, x_3, x_4\}$ and 4 attributes $A = \{B,C,E,D\}$. The attributes are partitioned into two types: condition and decision. The condition attributes are B, C, and E, and the decision attribute is D. They are written as,

$A_C = \{B, C, E\}$ *condition attribute*

$A_D = \{D\}$ *decision attribute*

The following notations are used for attribute values.

$B(x_1) = b_1$	*b_1 is the value of B in x_1*
$V_D = \{d_1, d_2\}$	*attribute values for attribute D*

We assume that the classification is guided by the decision values. Therefore, a class is a set of objects in *S* with the same decision value, which is written as,

$X_d = \{x_k \in X; D(x_k) = d\}$ *Class defined by decision value d*

In Table 1, $X_{d1} = \{x_1\}$ and $X_{d2} = \{x_2, x_3, x_4\}$. We use an information system that has two decision values (d_1 and d_2) for simplicity of presentation, but the algorithm directly carries over to the general case where $|V_D| \geq 2$.

2.2 Method Description

2.2.1 Differentiators

The *differentiator* used in this paper is a derivation of reduct [2] in rough set theory [1]. An o*bject-object differentiator,* denoted as $\Delta_{OO}(x',x'')$, is the disjunction of the attribute values in object *x'* that distinguishes *x'* from the other object *x''*. For example, in Table 1, we can easily distinguish x_1 from x_2 with b_1 of *B*. i.e., a person wearing a red shirt (b_1) from another person wearing a blue shirt (b_2). We do not need other attribute values to distinguish x_1 from x_2. $\Delta_{OO}(x_1, x_2)$ is b_1 *or* e_1, and it is written as ($b_1 \vee e_1$). Although we have two attribute values, one is enough to discern x_1. An object may contain null values. We use two methods for handling the null value: (1) if an attribute value of *x''* is null the corresponding attribute value in *x'* becomes an element of $\Delta_{OO}(x', x'')$ because the null value in *x''* is different from the value in *x'*. (2) if an attribute value in *x'* is null we do not add the null to $\Delta_{OO}(x', x'')$ because a null is not a specific value that can distinguish *x'* from other objects.

An *object-class differentiator,* denoted as $\Delta_{OC}(x', X_{d''})$, is *the shortest terms* that distinguish *x'* from $X_{d''}$ (e.g. x_1 and $X_{d2} = \{x_2, x_3, x_4\}$). The shortest terms are acquired by finding the prime implicants [12] of all object-object differentiators (Δ_{OO}) between *x' and* $\{x \in X_{d''}\}$. A prime implicant is a product term that cannot be subsumed by any other product term. To obtain the prime implicant set, we put all Δ_{OO}s into conjunctive normal form (CNF) and transform it to disjunctive normal forms (DNF). $\Delta_{OO} = \emptyset$ when two objects are identical. In this case, we do not include the empty set as a conjunct of the CNF. The object-class differentiator (Δ_{OC}) consisting of at least one empty Δ_{OO} is called approximated object-class differentiator. Otherwise, it becomes a precise object-class differentiator. When *x'* is compared with all other objects in $\{X - x'\}$ it is simply written as $\Delta_{OC}(x')$. Intuitively, $\Delta_{OC}(x')$ is the attribute values in *x'* that can most effectively (not necessarily precisely) differentiate *x'* from the other objects in *S*.

A *class-class differentiator*, denoted as $\Delta_{CC}(X_{d'}, X_{d''})$, is the disjunction of Δ_{OC}s. It is the set of unique terms in $X_{d'}$ in relation to $X_{d''}$. A $\Delta_{CC}(X_{d'}, X_{d''})$ is categorized as an approximated class-class differentiator if it has one or more approximated Δ_{OC}. Otherwise, it becomes a precise class-class differentiator. Some terms may appear more than once in a Δ_{CC}. The frequency of a term *t* is defined as the number of times *t* is a subset of *x* in X_d

$$f(t) = \sum_{i=1}^{n} \begin{cases} 1, & t \subseteq x_i : x_i \in X_d \\ 0, & otherwise \end{cases}$$

When $X_{d'}$ of $\Delta_{CC}(X_{d'}, X_{d''})$ is X - $X_{d''}$ (e.g. there are only 2 class labels), the class differentiator of $X_{d'}$ is simply written as $\Delta_{CC}(X_{d'})$. Let (t_i ,f_i) be a term and its frequency. Then, $\Delta_{CC}(X_{d'})$ is,

$$\Delta_{CC}(X_{d'}) = \{(t_1, f_1), (t_2, f_2),\ldots, (t_n, f_n)\}$$

We use $\Delta_{CC}(X_d)$ to classify a test object.

2.2.2 Classifying a test object

Suppose that we want to classify the object x_{new}. The class label of x_{new} is determined by its unknown decision value. We calculate the value by measuring how similar x_{new} is to the terms in the class differentiators. For example, the decision value of x_{new} is d_1 if x_{new} is closer to the terms of $\Delta_{CC}(X_{d1})$ than those of $\Delta_{CC}(X_{d2})$. The idea is that $\Delta_{CC}(X_{d1})$ is the distinctive pattern of values that characterizes the class. If the same pattern is found in x_{new}, x_{new} is most likely having the same characterestics of $\Delta_{CC}(X_{d1})$. Since the example problem has two class differentiators, we measure the similarity between the terms of x_{new} and the terms of $\Delta_{CC}(X_{d'})$ and $\Delta_{CC}(X_{d''})$ respectively. Then, we compare the degree of similarity to choose the right decision value. The degree of similarity, expressed as a weight, is calculated by the ratio between the sum of the frequencies of the terms in $\Delta_{CC}(X_d)$ and the number of the objects in X_d. Let $\omega^{x_{new}}(d)$ be the weight of the unknown decision value of x_{new}. Then,

$$\omega^{x_{new}}(d) = \frac{\sum f(t_i) : t_i \subseteq x_{new}, t_i \in \Delta_{CC}(X_d)}{|X_d|}$$

Next, we compare the weights of all decision values. The object x_{new} belongs to a class that has the highest $\omega^{x_{new}}$.

$$x_{new} \in X_d : \omega^{x_{new}}(d) = max(\omega^{x_{new}}(d_i)), i = 1 \ldots |V_D|$$

When two or more decision values have the equal weight, we randomly select one. We show an example in the next section.

2.3 Sample Problem

We will use the information system in Table 2. $A_C = \{B,C,E\}$. $A_D = \{D\}$. The dashed line indicates the division between two classes. $X_{d1} = \{x_1, x_2, x_3, x_4\}$ and $X_{d2} = \{x_5, x_6, x_7\}$. x_{new} is the object to be classified.

Table 2: Information System S and an object x_{new}

	B	C	E	D
x_1	b_2	c_1	e_1	d_1
x_2	b_2	c_3	e_1	d_1
x_3	b_1	c_1		d_1
x_4	b_1	c_3	e_1	d_1
x_5	b_1	c_1	e_1	d_2
x_6	b_1	c_1	e_1	d_2
x_7	b_2	c_2	e_2	d_2
x_{new}	b_2	c_3	e_1	

We first build a discernibility matrix from Table 2 in order to obtain $\Delta_{CC}(X_{d1})$. Table 3 is the discernibility matrix that the elements in each cell are the *object-object differentiators.* For example, b_2 is the attribute value that discerns x_1 from x_5 in Table 2. Then, b_2 is placed in the cell between x_1 and x_5 in Table 3. Either c_1 or e_1 can distinguish x_1 from x_7 (*or* is denoted as $\vee$ sign), and they are placed between x_1 and x_7 in Table 3.

Table 3: Discernibility matrix for X_{d1}

	x_1	x_2	x_3	x_4
x_5	b_2	$b_2 \vee c_3$		c_3
x_6	b_2	$b_2 \vee c_3$		c_3
x_7	$c_1 \vee e_1$	$c_3 \vee e_1$	$b_1 \vee c_1$	$b_1 \vee c_3 \vee e_1$

The conjunction of all $\Delta_{OO}(x_1)$s is $(b_2) \wedge (b_2) \wedge (c_1 \vee e_1)$. We transform it to a DNF to find $\Delta_{OC}(x_1)$. That is,

$\Delta_{OC}(x_1) = (b_2) \vee (b_2) \vee (c_1 \wedge e_1) = (b_2 \wedge e_1) \vee (b_2 \wedge c_1)$

We also compute Δ_{OC} for x_2, x_3, x_4.

$\Delta_{OC}(x_2) = (b_2 \wedge c_3) \vee (b_2 \wedge c_3) \vee (c_3 \wedge e_1) = (c_3) \vee (b_2 \wedge e_1)$

$\Delta_{OC}(x_3) = (b_1 \wedge c_1)$

$\Delta_{OC}(x_4) = (c_3) \wedge (c_3) \wedge (b_1 \wedge c_3 \wedge e_1) = (c_3)$

Next, we calculate the frequency of the terms in Δ_{OO}s to build $\Delta_{CC}(d_1)$. For instance, the frequency of c_3 is 2 because it is in $\Delta_{OC}(x_2)$ and $\Delta_{OC}(x_4)$. We can find $b_2 \wedge e_1$ twice in $\Delta_{OC}(x_1)$ and $\Delta_{OC}(x_2)$. Its frequency is 2. Thus, $\Delta_{CC}(d_1)$ *is,*

$\Delta_{CC}(d_1) = \{(c_3,2), (b_2 \wedge e_1,2), (b_2 \wedge c_1,1), (b_1,1), (c_1,1)\}$

The class-class differentiator for d_2, $\Delta_{CC}(d_2)$, can be obtained from the same matrix because there are only 2 decision values in *S* and the discernibility matrix is symmetric. Using the same method we acquire,

$\Delta_{CC}(d_2) = \{(b_1 \wedge c_1 \wedge e_1,2), (c_2,1), (e_1,1)\}$

Table 4 shows the terms, frequencies, and class labels of $\Delta_{CC}(d_1)$ and $\Delta_{CC}(d_2)$. We calculate the weight of the class label of x_{new} using Table 4. A term in Table 4 is counted to compute the weight if the term is a subset of x_{new}. We can find that term #1 = $\{c_3\}$ and term #2 = $\{b_2, e_1\}$ are the subsets of $\{b_2,c_3,e_1\}$. Their frequencies are 2 and 2 respectively. There are 4 objects in X_{d1}. Therefore, the weights of d_1 for x_{new} is,

$$\omega^{x_{new}}(d_1) = \frac{2+2}{4}$$

In the same way, term #8 is a subset of x_{new}, and we use its frequency value 1 and the number of terms in X_{d2} to calculate the weight of d_2.

$$\omega^{x_{new}}(d_2) = \frac{1}{3} = 0.33$$

We classify x_{new} to X_{d1} because $\omega^{x_{new}}(d_1) > \omega^{x_{new}}(d_2)$.

Table 4. Class differentiator (ΔCC) for S

term #	*term*	*frequency*	*class label*
1	c_3	2	d_1
2	$b_2 \cdot e_1$	2	
3	$b_2 \cdot c_1$	1	
4	b_1	1	
5	c_1	1	
6	$b_1 \cdot c_1 \cdot e_1$	2	d_2
7	c_2	1	
8	e_1	1	

3 Implementation and experiment

We implemented the algorithm in Python programming language and tested it using the data in Table 5. The data set has information about stolen cars. The training data has 10 objects $\{x_1,...,x_{10}\}$, three condition attributes {*color, type, origin*}, and a decision attribute {*stolen*}. It is divided into two classes by *'yes'* and *'no'*. All attributes are categorical.

Table 5. Stolen Car

object	***color***	***type***	***origin***	***stolen***
x_1	*red*	*sports*	*domestic*	*yes*
x_2	*red*	*sports*	*domestic*	*no*
x_3	*red*	*sports*	*domestic*	*yes*
x_4	*yellow*	*sports*	*domestic*	*no*
x_5	*yellow*	*sports*	*imported*	*yes*
x_6	*yellow*	*suv*	*imported*	*no*
x_7	*yellow*	*suv*	*imported*	*yes*
x_8	*yellow*	*suv*	*domestic*	*no*
x_9	*red*	*suv*	*imported*	*no*
x_{10}	*red*	*sports*	*imported*	*yes*

We conducted two experiments. In the first, we compared our algorithm to Naive Bayes classifier by running two algorithms with several different test objects. For example, both algorithms classified x_{new1} = {*red, suv, domestic*} to *'no'* as shown in Table 6. However, Naive Bayes classifier failed to classify another object x_{new2} = {*blue, suv, domestic*} to a class due to the zero probability problem created by the attribute value *blue*. We need to use some type of data modification, such as Laplace correction [3] to solve this problem. On the other hand, our algorithm successfully classified x_{new2} using the class differentiators listed below.

```
Class-Class Differentiator

Decision Value : Yes (5), Hit 0/5

(red ∧ sports)             2

(domestic ∧ red)           2

(imported ∧ sports)        2

(imported ∧ yellow)        1
```

```
Decision Value : No (5), Hit 3/5

(domestic)                    1

(domestic ∧ yellow)           2

(suv)                         1

(domestic ∧ suv)              1

(red ∧ suv)                   1
```

In Table 5, the attribute value set {*suv, domestic*} is found only in x_8, and it works as a class differentiator of X_{no}. This term is a subset of x_{new2} = {*blue, suv, domestic*}, and is used to predict the decision value of x_{new2} to '*no*'.

Table 6. Classification of x_{new1} and x_{new2}.

Algorithm	x_{new1}	x_{new2}
Naive Bayes	*yes : 0.037*	*yes : 0.0*
	no : 0.069	*no : 0.0*
Class Differentiator	*yes : 0.4*	*yes : 0.0*
	no : 0.8	***no : 0.6***

We also compared our algorithm to a rule based classifier. Although the details vary, most rule based classifiers generate a set of if-then rules to predict the unknown data [11]. As described earlier, we often need to run a rule extraction algorithm several times to obtain the rule set that can classify an object. In this experiment, we extracted two sets of rules using ERID [14] and ran Chase [13] algorithm to classify x_{new2} = {*blue, suv, domestic*}. As shown in Table 7, we could classify x_{new2} to X_{no} using ERID rule set #1 (that is generated with support = 3 and confidence = 0.75). However, no decision value was predicted with another rule set (rule set #2 with support = 2 and confidence = 0.8) because the terms in rule set #2 did not have a matching attribute value. The proposed algorithm does not have this problem because the class differentiators are not dependent on input parameters such as threshold values.

```
Rules generated by ERID

[Rule set #1] min support 3, confidence 0.75

suv→No                        3, 0.75

red,sports→Yes                3, 0.75

[Rule set #2] min support 2, confidence 0.8

yellow,domestic→No            2, 1.0

sports,imported→Yes           2, 1.0
```

Table 7. Classification of x_{new2} using Chase

	sup =3, conf = 0.75	*sup = 2, conf = 0.8*
Rule Based Classification	*yes : 0.0*	*yes : 0.0*
	no : 1.0	*no : 0.0*

4 Conclusion

This paper discussed a method for the supervised classification of categorical data. The algorithm uses class differentiators to find out the class label of a test object. The class differentiators are the set of attribute values in the training data set that distinguishes one class from the others. The proposed algorithm measures the similarity between the test object and the class differentiators to determine the class membership. The comparison of our algorithm with a rule based classifier and Naive Bayes classifier shows that the proposed algorithm produces more consistent results and less prone to the zero probability problem.

REFERENCES

[1]. Pawlak, Z., "Rough sets", International Journal of Computing and Information Sciences, 11(5), pp. 341-356, 1982

[2]. Pawlak, Z., Skowron, A., "Rough sets and Boolean reasoning" Information Sciences, 177(1), pp. 41–73, 2007

[3]. Duda, R., Hart, P. and Stork, D., "Pattern classification", Wiley, New York, 2nd edition, 2001

[4]. Tan, P., Steinbach, M. and Kumar, V., "Introduction to data mining", 1st edition, Pearson Addison Wesley, Boston, 2005

[5]. Boriah, S., Ch, ola, V. and Kumar, V., "Similarity measures for categorical data: A comparative evaluation", In: SIAM Data Mining Conference, pp.243-254, 2008

[6]. Dougherty, J., Kohavi, R., Sahami, M., "Supervised and unsupervised discretization of continuous features", In: Twelfth International Conference on Machine Learning. pp.194-202, 1995

[7]. Guarino, N., "Formal ontology, conceptual analysis and knowledge representation", International journal of human-computer studies, 43(5), pp.625-640, 1995

[8]. Xiaodan Zhang, Liping Jing, Xiaohua Hu, Michael Ng, and Xiaohua Zhou. "A comparative study of ontology based term similarity measures on PubMed document clustering", In: 12th international conference on Database systems for advanced applications, pp.115-126, 2007

[9]. Brickell, J. and Shmatikov, V., "The cost of privacy: destruction of data-mining utility in anonymized data publishing" In: ACM SIGKDD international conference, pp.70-78, 2008

[10]. Im, S., "Privacy aware data management and chase", Fundamenta Informaticae, 78(4), pp.507-524, 2007

[11]. Charu C. Aggarwal, and ChengXiang Zhai., "A Survey of Text Classification Algorithms", Mining Text Data, Springer, pp 43-76, 2012

[12]. Posthoff, Ch. and Steinbach, B., "Logic Functions and Equations - Binary Models for Computer Science", Springer, Dordrecht, 2004

[13]. Dardzinska, A. and Ras, Z., "Rule-based Chase algorithm for partially incomplete information systems", In: Second International Workshop on Active Mining, pp.42-51, 2003

[14]. Dardzinska, A. and Ras, Z.W., "Extracting Rules from Incomplete Decision Systems: System ERID" In: Foundations and Novel Approaches in Data Mining, Studies in Computational Intelligence, Vol. 9, Springer, pp.143-154, 2006