An Instance-Based Learning Method for Databases: An Information Theoretic Approach

Changhwan Lee

Department of Computer Science and Engineering
University of Connecticut, Storrs, CT 06269

Abstract. A new method of instance-based learning for databases is proposed. We improve the current similarity measures in several ways using information theory. Similarity is defined on every possible attribute type in a database, and also the weight of each attribute is calculated automatically by the system. Besides, our nearest neighbor algorithm assigns different weights to the selected instances. Our system is implemented and tested on a typical machine learning database.

1 Introduction

Instance-based learning(IBL) algorithms have shown high classification accuracies, and its power has been demonstrated in a number of important real world domains (e.g. [2] [5]). As each instance is represented by a set of attribute-value pairs, instance-based learning algorithms, which learn by storing examples as points in a feature space, require some means of measuring similarity between instances [1]. When the feature values are numeric, Euclidean distance can be used to compare examples. However, in a database environment, there exist other types of attributes including categorical or pointer type. Due to this problem, a more sophisticated treatment of the feature space is required. Recently there have been several attempts to define similarity for categorical domains [3] [7]. However, these methods can not be used directly in the database environment either due to the lack of the similarity measure for pointer type or due to the lack of the weights for attributes. In this paper, we point out some problems that current similarity measures contain and propose a new improved method of measuring similarity between instances in databases. We define similarity measures for every attribute type in the database, and provide methods to calculate the weights for both attributes and selected instances. Our method has been implemented and tested on a typical machine learning database. Our classification accuracy is compared with that of other well known methods.

2 Information Theoretic Similarity Measure

We improve the current similarity measures in a number of ways. Firstly, we take into account the weights of attributes, and the weights will be calculated automatically by the system. Secondly, our learning program handles variables

which take any data type: binary, continuous, categorical, or pointer. Pointer types are different from categorical types with respect to the way they store information. With current similarity measure, we would lose part of the information given by pointer types. Thirdly, some stored instances are more reliable classifier than others. Intuitively, one would like these trustworthy examplars to have more drawing power than others. We accomplish this by giving reliable instances higher values, making them appear closer to a new example.

Now we will describe the detailed method of calculating similarity metrics. Suppose we compare the similarity between two instance X and Y, and let x_i and y_i, for $i = 1, \ldots, k$, be the values of i-th attribute for X and Y, respectively. Let a_i be a value of an attribute A, and its target attribute is denoted as T. Let $\Theta_T(X, Y)$ denote the similarity function between X and Y with respect to attribute T. $\Theta_T(X, Y)$ will be defined in the following manner.

$$\Theta_T(X, Y) = \omega_1 \Theta(x_1, y_1) + \omega_2 \Theta(x_2, y_2) + \cdots + \omega_k \Theta(x_k, y_k) \tag{1}$$

where ω_i is the weight of attribute A_i, and $\Theta(x_i, y_i)$ denotes the similarity between values x_i and y_i. The computation of similarity between two instances consists of two steps: calculating the weights of attributes and calculating value similarity between two attribute values. Each step will be explained in detail in the following.

The basic idea for calculating the weight of each attribute is that the more information an attribute gives to the target attribute, the more weight the attribute is to have. For one value assignment of the attribute, its corresponding information content will be calculated using entropy function. For discrete attribute types such as binary, categorical or pointer, we calculate the information content for each separate discrete value and their average value becomes the weight of the attribute. For numeric attributes, we assume that the values are discretized in advance, and the same procedure will be applied. Now the critical part is how to measure the amount of information a value assignment gives to the target attribute. Suppose we are to estimate the amount of information that a value assignment of an attribute A gives to the target attribute T. Kullback [4] has proposed the following measure for the information content of a value assignment.

$$Ent(T|A = a) = \sum_t p(t|a) \cdot \log \frac{p(t|a)}{p(t)} \tag{2}$$

where t, a represent value assignments of T, A, respectively. This measure is interpreted as the average mutual information between the events t and a with the expectation taken with respect to the a posteriori probability distribution of T. It can be viewed as a similarity between two probability distributions. Summation of formula (2) with respect to a may serve as the weight of attribute A. However, in that case, attributes with a large number of category are to have larger weights since the weights increase monotonically as the number of category increases. To avoid this problem, formula (2) is multiplied by the probability that each category can happen, $p(a)$. Furthermore, because the value of this measure can grow indefinitely, using $Ent(T|A)$ as the weight of an attribute may provide

anomalous similarity values. By dividing the current $Ent(T|A)$ value by the total summation of $Ent(T|A)$, we have the following formula which ranges from zero to one. Therefore, the final formula for an attribute A is given as

$$\frac{Ent(T|A)}{\sum_A Ent(T|A)} = \frac{\sum_a p(a)\, Ent(T|A=a)}{\sum_A Ent(T|A)} \quad . \tag{3}$$

The second step for calculating instance similarity is to compute the similarity between two attribute values. In the following, a new method for calculating similarity between two values, $\Theta(x_i, y_i)$ of equation 1, is defined for each data type: binary, categorical, numeric, and pointer.

For binary values, we just check whether they match with each other.

$$\Theta(x_i, y_i) = \begin{cases} 1 & : \quad x_i = y_i \\ 0 & : \quad x_i \neq y_i \end{cases} \tag{4}$$

For categorical(nominal) values, we first measure the amount of information that x_i or y_i gives about the target attribute. Formally, the similarity is given as

$$\Theta(x_i, y_i) = 1 - \frac{|Ent(T|A=x_i) - Ent(T|A=y_i)|}{Ent(T|A)} \quad . \tag{5}$$

For numeric values, the distance between values x_i and y_i is defined as the ratio of absolute value of $|x_i - y_i|$ to the total range of A. Therefore, the similarity between these two values is defined as

$$\Theta(x_i, y_i) = 1 - \frac{|x_i - y_i|}{|A_{max} - A_{min}|} \tag{6}$$

where A_{max}, A_{min} denote the maximum and minimum values of attribute A, respectively.

For pointer attribute type, since a value of pointer type refers to another instance, the difference between two pointer values can be regarded as the difference between those two which are being referred to. The formal definition of similarity for pointer value is defined in recursive form as follows: In case either or both of x_i, y_i refer to instances which does not exist in the database, the similarity is defined as zero. Otherwise,

$$\Theta(x_i, y_i) = \begin{cases} \Theta_T(x_i, y_i) & : \quad x_i \neq y_i \\ 1 & : \quad x_i = y_i \end{cases} \tag{7}$$

3 Algorithm and Experimental Results

Our instance-based learning algorithm stores a series of training instances in its memory, and uses a similarity metric to compare new instances to those stored. The algorithm requires two passes of the training set. During the first pass, the weights of feature will be calculated. The second pass is to calculate a standard similarity metric for measuring the similarity between two examples in a multi-dimensional feature space. Another different feature of our system from other

instance-based learners is that we give different weights to the selected instances. In our system, closer instances are assigned larger weight values. Among the instances selected from the database, the similarity values of instances with the same class value will be accumulated, and the class value which has the highest collected similarity values will be the final classification value.

Tic-tac-toe game database has been selected for the purpose of tesing our algorithm. It encodes the complete set of possible board configurations at the end of tic-tac-toe games. Each of the nine attributes in the database corresponds to one tic-tac-toe square. It contains 958 instances of legal tic-tac-toe endgame boards. Figure 1 shows the attributes and their corresponding weights calculated by applying our algorithm. As we can see in Figure 1, the attribute number 5 has the highest weight among others, which is intuitively correct. When we play the tic-tac-toe game and we mark the center square in the first place, we are supposed to win ! In addition, we can see that the squares of 2, 4, 6, and 8 have the same weight, which is correct because they are symmetric. Similarly, the weights of 1, 3, 7, and 9 also show the same weight as well. The results of other algorithms are shown in Figure 2. Our system has shown excellent performance in this database.

0.0800	0.0413	0.0800
(1)	(2)	(3)
0.0413	0.5145	0.0413
(4)	(5)	(6)
0.0800	0.0413	0.0800
(7)	(8)	(9)

Fig. 1. Weights of tic-tac-toe attributes

ID3	CN2	IB3	Proposed Method
84.0%	98.1%	99.1%	99.5%

Fig. 2. Success rates for tic-tac-toe data

References

1. D. Aha, D. Kibler and M. Albert Instance-based Learning Algorithms. *Machine Learning*, 6(1) pp. 37-66, 1991.
2. L. W. Bradshaw. Learning about speech sounds: The NEXUS project. *Proceedings of the Fourth Int'l Workshop on Machine Learning*, Irvine, CA: Morgan Kaufmann, pp. 1-11, 1987.
3. S. Cost and S. Salzberg. A Weighted Nearest Neighbor Algorithm for Learning with Symbolic Features, *Machine Learning*, Vol. 10, pp. 57-78, 1993.
4. S. Kullback, *Information Theory and Statistics*, New York: Dover Publications, 1968.
5. J. M. Kurtzberg. Feature analysis for symbol recognition by elastic matching. *IBM Journal of Research and Development*, 31:91-95, 1987.
6. P. Smyth and R. M. Goodman. Rule Induction Using Information Theory, in G. P. Shapiro and W. J. Frawley, editor, *Knowledge Discovery in Databases*, MIT Press, 1991.
7. C. Stanfill and D. Waltz. Toward Memory-based Reasoning, *Communications of the ACM*, 29(12), pp. 1213-1228, 1986.