CODE SLICING TO IMPROVE CASE CLASSIFICATION ACCURACY

Omar Abdalgani Shiba[1*], Mohamed Nasir Sulaiman[1], Fatimah Ahmad[1] and Ali Mamat[1]

Received: May 30, 2003; Revised: Mar 1, 2004; Accepted: Apr 4, 2004

Abstract

Finding a good classification algorithm is an important component of many data mining projects. Data mining researchers often use classifiers to identify important classes of objects within a data repository. The goal of this paper is to improve the case classification accuracy in data mining. The paper achieves this goal by introducing a new approach of similarity-based retrieval based on program slicing techniques and is called Case Slicing Technique (CST). The proposed approach helps identify the subset of features used to compute the similarity measures needed by the classification algorithms. The idea is based on slicing cases with respect to the slicing criterion. Likewise, the paper presents the experimental results of the CST using five real-world datasets which are; Australian Credit Application (AUS), Cleveland Heart Disease (CLEV), Breast Cancer (BCO), German Credit Card (GERM) and Hepatitis Domain (HEPA). The paper compares CST with other selected approaches. The results obtained showed that the classification accuracy can be improved when we use CST.

Keywords: Data mining, case-slicing technique, classification accuracy, case classification

Introduction

One of the problems addressed by machine learning is that of data classification. Since the 1960ís, many algorithms for data classification have been proposed (Thamar and Olac, 2001). The problem of classification is defined as follows: The input data is referred to as the training set, which contains a plurality of records, each of which contains multiple attributes or features. Each example in the training set is tagged with a class label. The class label may either be categorical or quantitative. The problem of classification in the context of a quantitative class label is referred to as the regression-modeling problem. The training set is used in order to build a model of the classification attribute based upon the other attributes. This model is used in order to predict the value of the class label for the test set. Some well-known techniques for classification include the following: K-Nearest Neighbor (K-NN) (Wettschereck and Aha, 1995; Xiaoli, 1999), Bayesian Learners such as Naïve Bayes (NB) (Michell, 1997) and Base Learning Algorithm (C4.5) (Quinlan, 1986,1993; Thamar and Olac, 2002). This paper introduces a new classification approach based on the program slicing techniques which help identify the subset of features used to compute the similarity measures (needed by the classification algorithm). The slicing technique was proposed for procedural

[1] *Faculty of Computer Science and Information Technology University Putra Malaysia 43400 UPM Serdang, Selangor E-mail : abumoad99@hotmail.com*

[*] *Corresponding author*

programming languages (Weiser, 1984). Slicing is a method used by experienced computer programmers for restricting, the behaviour of a program to some specified subset of interest. Several slicing algorithms for imperative languages have been developed. The slicing of the programs is performed with respect to some criteria; Weiser proposes as a criterion the number *i* of a command line and a subset *V* of program variables. According to this criterion, a program is analyzed and its commands are checked for their relevance to the command line *i* and those variables in *V*. However, other authors have defined different criterion (Tip, 1995; Vasconcelons, 2000).

This paper investigates the possibility of using the slicing technique successfully with classification problem and the comparison between the slicing technique as a classification approach and *K-NN*, Naïve Bayes and C4.5 is presented.

The rest of the paper is organized as follows: the next section presents some related work. Section 3 describes the case slicing technique. Section 4 presents the experimental results that compare the performance of the proposed approach and other selected algorithms. Finally the conclusion is presented in section 5.

Related Work

In this section, three selected classification algorithms related to the proposed approach are briefly described.

K-Nearest Neighbor (K-NN)

The basic idea of the K-Nearest Neighbor algorithm *(K-NN)* is to compare every attribute of every case in the set of similar cases with every corresponding attribute of the input case. A numeric function is used to decide the value of comparison. Then the *K-NN* algorithm selects a case with the highest comparison value and retrieves it (Xiaoli, 1999).

The *K-NN* assumes that each case X= $\{x_1, x_2 \ldots x_n, x_c\}$ is defined by a set of *n* (numeric or symbolic) features, where x_c is *x's* class value. Given a query *q* and a case library *L, K-NN* retrieves the set *k* of *q's k* most similar (i.e., the least distant) cases in *L* and predicts their weighted majority class as the class of *q* (Wettschereck and Aha, 1995). Distance in *K-NN* is defined as in equation (1).

$$distance\ (x,q) = \sqrt{\sum_{f=1}^{n} w_f^{\,*difference\ (xf,qf)^2}} \qquad (1)$$

where w_f is the parameterized weight value assigned to feature f as in equation (2).

$$w_f = P\left(C|i_a\right) \qquad (2)$$

That is, the weight for feature *a* for a class *c* is the conditional probability that a case is a member of *c* given the value to *a* where $P(C|i_a)$ is defined in equation (3); and the difference between *x* and *q* can be calculated as in equation (4).

$$P\left(C|i_a\right) = \frac{|instances\ containing\ i_a\ class\ =\ C|}{|instances\ containing\ i_a|} \qquad (3)$$

$$difference\left(x_f, q_f\right) = \begin{cases} |x_f - q_f| & if\ feature\ f\ is\ numric \\ 0 & if\ feature\ f\ is\ symbolic \quad \& x_f = q_f \\ 1 & otherwise \end{cases} \qquad (4)$$

In equation (2) *K-NN* assigns equal weights to all features (i.e. $\forall f\{wf = 1\}$).

Naïve Bayes Classifier (NB)

The estimates of the probability masses are used as an input for the Naive Bayes classifier. This classifier simply computes the conditional probabilities of the different classes giving the values of attributes and then selects the class with the highest conditional probability. If an instance is described with *n* attributes a_i $(i=1...n)$, then the class of that instance is classified to a class *v* from a set of possible classes *V* according to a Maximum A Priori criterion (MAP), the Naïve Bayes classifier can be defined as in equation (5).

$$v = \arg\max_{v_j \in V} p(v) \prod_{i=1}^{n} p(a_i|v_i) \qquad (5)$$

The conditional probabilities in the above formula are obtained from the estimates of the probability mass function using the training data. This Bayes classifier minimizes the probability of a classification error under the assumption that

the sequence of points is independent (Michell, 1997; Thamar and Olac, 2001).

The Base Learning Algorithm (C4.5)

C4.5 is an extension to the decision-tree learning algorithm ID3 (Quinlan, 1986,1993; Thamar and Olac, 2002). Only a brief description of the method is given here and more information can be found in (Quinlan, 1993). The algorithm consists of the following steps:

1. Build the decision tree from the training set (conventional ID3).
2. Convert the resulting tree into an equivalent set of rules. The number of rules is equivalent to the number of possible paths from the root to a leaf node.
3. Prune each rule by removing any preconditions that result in improving its accuracy, according to a validation set.
4. Sort the pruned rules in descending order according to their accuracy, and consider them in this sequence when classifying subsequent instances.

Description of the Proposed Approach

In this section the proposed classification approach and some related terms are discussed.

Program Slicing Technique

A slice is constructed by deleting those parts of the program that are irrelevant to the values stored in the chosen set of variables at the chosen point. The point of interest is usually identified by annotating the program with line numbers which identify each primitive statement and each branch node (Weiser, 1984; Horwitz and Reps, 1990). Program slicing is useful for program understanding, maintenance, debugging, testing, differencing, specialization, reuse, optimization, parallelization, and anomaly detection (Gallagher and Lyle, 1991). Program slicing has been widely studied in the context of imperative programs. Several slicing algorithms for imperative languages have been developed (Tip, 1995). Slicing of programs is performed with respect to some criteria. Weiser (1984) proposes as a criterion the number i of a command line and a subset V of program variables. According to this criterion, a program is analyzed and its commands are checked for their relevance to command line i and those variables in V. However, other authors have defined different criteria (Tip, 1995; Vasconcelons, 2000). Program slicing can be summarized as follows:

Program Slice ≡
- The statements (and predicates) that might affect the value of a set of variables at a particular statement.
- A slice is taken with respect to a set of variables at a particular statement, the *slicing criterion*.
- A slice and the actual program behavior are identical.

Executable (Slice) ≡ a slice that can be compiled and executed.

Closure (Slice) ≡ an informational presentation of a slice that might lack semantics.

Basic Types:
- Static vs. Dynamic
- Type of feedback: executable, closure
- Approach: graph reachability, dataflow equations using the control flow.

Extending Program Slicing to Case Slicing

The case slicing technique described in this paper addresses the problem of classification. The case slicing technique is an extension of program slicing technique. When we slice a case, we are interested in automatically obtaining that portion 'features' of the case responsible for specific parts of the solution of the case at hand. Some basic definitions of the case slicing term are:

A Case Slicing is a process for automatically obtaining subparts (features) of a case with a collective meaning.

A Slicing Criterion denotes the conditions of the slice computation, with respect to which and for which case a slice is required.

Sliced Case contains all features that could have direct relations with the features of interest at new case.

The Basic Idea

Conceptually, the proposed method is a

variation of the Nearest Neighbor Algorithms (Wettschereck and Aha, 1995; Wettschereck and Thomas, 1995; Xiaoli, 1999) and is called Case Slicing Technique (CST). It compares new cases with the training cases in the data file. Likewise, it computes the similarity between the new cases and the training cases to classify the new cases. The proposed method is a classification technique based on slicing. Slice case means we are interested in automatically obtaining that portion "features" of the case responsible for specific parts of the solution of the case at hand. By slicing the case with respect to the important features, we can obtain a new case with a small number of features or with only the important features. The proposed approach consists of a database with three calculation modules as follows:

Features Weighting Module

This module is used to measure the importance of each attribute in classification. The weight of each attribute has been calculated to classify the new case by using simple conditional probabilities. High weight values were assigning to features that are highly correlated with the given class using equations (2, 3) above. Where the weight for feature *a* for a class *c* is the conditional probability that a case is a member of *c* given the value to *a*

Discretization Computing Module

Discretization as used in this paper, and in the machine learning literature in general, is a process of transforming a continuous attribute values into a finite number of intervals and associating with each interval a discrete, numerical value. The usual approach for learning tasks that use the mixed-mode (continuous and discrete) data is to perform discretization prior to the learning process (Catlett, 1991; Fayyad and Irani, 1992; Dougherty *et al.,* 1995; Pfahringer, 1995).

The discretization process finds the number of discrete intervals, and then the width, or the boundaries for the intervals, given the range of values of a continuous attribute. Very often the user must specify the number of intervals, or provide some heuristic rules to be used (Ching *et al.,* 1995). A variety of discretization methods have been developed in recent years. Some models that have used the Value Difference Metrics (VDM) or variants of it (Cost and Salzberg, 1993; Rachlin *et al.,* 1994; Mohri and Tanaka, 1994) have discretized continuous attributes into a somewhat arbitrary number of discrete ranges, and then treated these values as nominal (discrete unordered) values.

When using the slicing approach, continuous values are discretized into *s* equal-width intervals (though the continuous values are also retained for later use), where *s* is an integer supplied by the user. Unfortunately, there is currently little guidance on what value of *s* to use. Current research is examining more sophisticated techniques for determining good values of *s*, such as cross-validation, or other statistical methods (Wilson and Martinez, 1996). The width w_a of a discretized interval for attribute *a* is given by equation (6).

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

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

The discretized value *v* of a continuous value *x* for attribute *a* is an integer from 1 to *s*, and is given by equation (7).

$$v = disc_a(x) = \begin{cases} \left\lceil \frac{(x - min_a)}{w_a} \right\rceil & \text{if attribute } a \text{ is continuous} \\ x & \text{if attribute } a \text{ is discrete} \end{cases} \tag{7}$$

Distance Computation Module

There are many learning systems that 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. A variety of distance functions are available for such uses, including the Minkowsky (Batchelor, 1978), Mahalanobis (Nadler and Eric, 1993), Camberra, Chebychev, Quadratic, Correlation, and Chi-square distance metrics (Edwin, 1974; Michalski *et al.,*1981), the Context-Similarity measure (Biberman,

1994), the Contrast Model (Tversky, 1977), hyperrectangle distance functions (Salzberg, 1991; Domingos, 1995) and others.

Although there are many distance functions being proposed, by far the most commonly used is the Euclidean distance function, which is defined in equation (8).

$$E(x,y) = \sqrt{\sum_{a=1}^{m}(x_a, y_a)^r} \quad (8)$$

Where x and 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 not often computed in practice because the closest instance(s) will still be the closest, regardless of whether the square root is taken or not.

An alternative function, the *City-block* or *Manhattan* distance function, requires less computation and is defined in equation (9).

$$M(x,y) = \sum_{a=1}^{m}\left|x_a - y_a\right|^r \quad (9)$$

The Euclidean and Manhattan distance functions are equivalent to the Minkowskian r-distance function (Batchelor, 1978) with $r = 2$ and 1, respectively.

Slicing Technique

The objective of the slicing technique is to optimize the similarity matching to achieve the best classification results. The proposed approach is adapting the slicing techniques that have been used in programming languages, to slice the cases by removing subset of features which are irrelevant to case label with respect to the selected slicing criterion. The case classification algorithm is shown in Figure 1.

A Formal Description of Case Slicing Technique

Below is a formal description of a basic Case Slicing Technique in order to have a detailed investigation of the approach.

Let

S = $\{C_1, C_2, C_3, ..., C_n\}$ *set of cases in Case Base* $\forall S \exists C_i \; S \neq \phi$

C_i = $\{f_1, f_2, f_3, ..., f_n\}$ where *n is the number of features in* C_i

λ = $[\{C_s | C_s$ *is a set of sliced cases*$\}]$ or

λ = {*all cases that contains one or more important feature(s)*}

I = $\{if_1, if_2, ..., if_n\}$ where *n is the number of important features in I*

$I \subseteq C_i \subseteq$ S

$I \subseteq C_s \subseteq \lambda$

```
Algorithm:   Algorithm for case Classification
Input:       User's Input Problem Specification
Output:      Classified Case
Begin
While true do
        Discrete_continuous_values ()
        { To convert continues values into discrete values}
        Assign_Weight_Cases()
        { To assign weights to each feature in the each acse}
        Slicing_Cases_w.r.t.Slicing_Criterion ()
        { To slice the cases with respect to important features }
        Calculate_Distance ()
        {To find the distance between each two feature in a
        case}
        Closer_Case_Searching ()
        { To find closest case to the case at hand}
        Return_Classified_Case ()
        { To assign class label to the new case }
Enddo
End.
```

Figure 1. Case classification algorithm based slicing.

Experimental Results

The classification algorithms intend to classify objects better both in terms of accuracy and speed. In most of the cases, accuracy is more important, as the aim of this paper we are focusing on the accuracy task, optimizations are carried out over the existing classification algorithms.

In this section the results of several practical experiments are presented to examine the performance of the proposed approach and other selected classification algorithms on five real-world problems.

Selected Datasets

In this paper, five real-world datasets which are widely used in the machine-learning field for evaluation of case slicing technique have been used. The five datasets: Australian Credit Card Approval (AUS), Cleveland Heart Disease (CLEV), Breast Cancer (BCO), German Credit Card (GERM) and Hepatitis Domain (HEPA) were chosen from the UCI: Machine Learning Repositories and Domain Theories (Murphy, 1996). Table 1 presents the main characteristics of these datasets, where *B, C* and *D* in the table mean Boolean, continuous and discrete attributes respectively.

Empirical Results

We evaluated the performance of the case slicing technique by comparing it with the K-NN, C4.5 and Naïve Bayes classifiers on a variety of datasets. The datasets we have selected are a very good choice to test and evaluate the slicing technique because the datasets are from different domains and there is a good mixture of continoues, discrete and Boolean features. In all the experiments reported here we used the evaluation technique 10-fold cross-validation (Kohavi, 1995), which consists of randomly dividing the data into 10 equally, sized subgroups and performing ten different experiments. We separated one group along with their original labels as the validation set; the other groups were considered as the starting training set; the remainder of the data were considered the test set. Each experiment consists of ten runs of the procedure described above, and the overall average has been reported here. The criterion of choosing the best classification approach is based on the highest percentage of classification. The results, given in Table 2, list the classification accuracies achieved by each approach for each of the datasets, and the differences in accuracy are shown in Figure 2.

Table 1. Characteristics of the selected datasets.

Datasets	No. of data	Type & no. of attributes	No. of classes
AUS	690	9D, 6C (15)	2
CLEV	303	9D, 6C (15)	2
BCO	699	13B, 6C (19)	2
GERM	1,000	7C, 13D (21)	2
HEPA	155	13B, 6C (19)	2

Table 2. The classification accuracy achieved by the different classification algorithms.

	Methods (%)			
Datasets	K-NN	C4.5	Naïve bayes	CST
AUS	81.90	84.50	84.90	99.30
CLEV	71.20	77.20	83.40	96.00
BCO	97.10	74.70	96.40	99.30
GERM	69.40	98.50	70.30	98.00
HEPA	92.90	80.80	86.30	97.00

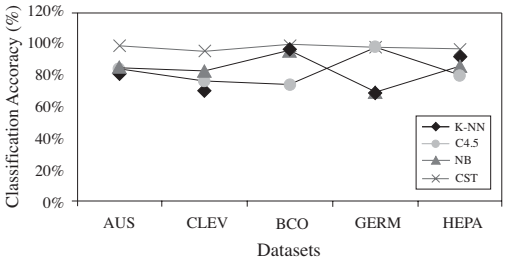

Figure 2. Comparison of the 10-fold cross-validated classification accuracies of the selected classification techniques.

Conclusion

The paper has presented and discussed the Case Slicing Technique (CST) as a new classification method to improve the case classification accuracy in data mining. The paper examines the technique on five real-world datasets. The results obtained showed that the classification accuracy can be improved when we use CST. The performance of the technique is compared with some other techniques and has possessed a competitive result and it gives a very high percentage of classification accuracy over other approaches.

References

Batchelor, B. (1978). Pattern recognition: ideas in practice. Plenum Press, New York, p. 71-72.

Biberman, Y. (1994). A Context similarity measure. Proceedings of the European Conference on Machine Learning (ECML-94); April 6-8, 1994. Catalina, Italy. Springer Verlag, p. 49-63.

Ching, J.Y., Wong, A.K., and Chan, K.C. (1995). Class-dependent discretization for inductive learning from continuous and mixed mode data. IEEE Transactions on Pattern Analysis and Machine Intelligence, 17(7):641-651.

Cost, S., and Salzberg, S. (1993). A weighted nearest neighbor algorithm for learning with symbolic features. Machine Learning, 10:57-78.

Domingos, P. (1995). Rule induction and instance-based learning: a unified approach. Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-95); August 20-11, 1995. Montreal, Canada; Morgan Kaufmann, p. 1,226-1,232.

Dougherty, J., Kohavi, R., and Sahami, M. (1995). Supervised and unsupervised discretization of continuous features. Proceedings of the 12th International Conference on Machine Learning; July 9-12, 1995. San Francisco, Tahoe City; CA: Morgan Kaufmann, p. 194-202.

Edwin, D. (1974). Recent progress in distance and similarity measures in pattern recognition. Proceedings of 2nd International Joint Conference on Pattern Recognition; July 4, 1974. Tokyo, p. 534-539.

Fayyad, U.M., and Irani, K.B. (1992). On the handling of continuous-valued attributes in decision tree generation. Machine Learning, p. 8:87-102.

Gallagher, K., and Lyle, J. (1991). Using program slicing in software maintenance. IEEE Transaction on Software Engineering, 17(8):751-761.

Horwitz, S., Reps, T., and Binkley, D. (1990). Interprocedural slicing using dependence graphs. ACM Transactions on Programming Languages and Systems, 12(1):35-46.

Kohavi, R. (1995). A Study of cross-validation and bootstrap for accuracy estimation and model selection. Proceedings of 14th International Joint Conference on Artificial Intelligence (IJCAI). San Mateo CA: Morgan Kaufmann, p. 1,137-1,145.

Michalski, R., Robert, E., and Edwin, D. (1981). A Recent advance in data analysis: clustering objects into classes characterized by conjunctive concepts. Progress in Pattern Recognition, 1, Kanal, L.N., and Rosenfeld, A. (eds.). North-Holland, New York, p. 33-56.

Mitchell, T.M. (1997). Machine learning. McGraw-Hill, New York.

Mohri, T., and Tanaka, H. (1994). An optimal-weighting criterion of case indexing for both numeric and symbolic attributes. David, W.A (ed.), Case-Based Reasoning: Papers from the 1994 Workshop (Report No. WS-94-01). Menlo Park, CA: AIII Press, p.123-127.

Murphy, P.M. (2001). UCI Repositories of Machine Learning and Domain Theories. University of California, Irvine UCI. Available from:http://www.isc.uci.edu/~mlearn/MLRepository.html. Accessed November 12, 2001.

Nadler, M., and Eric, P. (1993). Pattern recognition engineering. Wiley, New York. p. 293-294.

Pfahringer, B. (1995). Compression-based discretization of continuous attributes. Proceedings of the 12th International Conference on Machine Learning. US, Lake Tahoe, p. 456-463.

Quinlan, J.R. (1993). C4.5: programs for machine learning. CA: Morgan Kaufmann Publishers, Inc.

Quinlan, J.R. (1986). Induction of decision trees. Machine Learning, 1(1):81-106.

Rachlin, J., Simon, K., Salzberg, S., and David, W. (1994). Towards a better understanding of memory-based and bayesian classifiers. Proceedings of the Eleventh International Machine Learning Conference. New Brunswick, NJ: Morgan Kaufmann, p. 242-250.

Salzberg, S. (1991). A nearest hyperrectangle learning method. Machine Learning, 6:277-309.

Tapia, R., and Thompson, J. (1978). Nonparametric probability density estimation, Baltimore. MD: The Johns Hopkins University Press.

Thamar, S., and Olac, F. (2001). Improving classifier accuracy using unlabeled data. Proceedings of the IASTED International Conference on Artificial Intelligence and Applications (AIA2001). Marbella, Spain.

Thamar, S., and Olac, F. (2002). Improving classification accuracy of large test sets using the ordered classification algorithm. Proceedings of IBERAMIA-02, Lecture Notes in Computer Science, Sevilla Spain, Springer-Verlag Heidelberg, p. 70-79.

Tip, F. (1995). A Survey of program slicing techniques. Journal of Programming Languages, 3:121-189.

Tversky, A. (1977). Features of similarity. Psychological Review, 84(4):327-352.

Vasconcelos, W.W. (2000). Slicing knowledge-based systems techniques and applications. Knowledge-Based Systems Journal, 13:177-198.

Weiser, M. (1984). Program slicing. IEEE Trans. Software Engineering, 10(4):352-357.

Wettschereck, D., and Aha, D.W. (1995). Weighting features. Proceedings of the 1st. International Conference on CBR (ICCBR-95). Sesimbra, Portugal. Springer-Verlag, p. 347-358.

Wettschereck, D., and Thomas, G.D. (1995). An experimental comparison of nearest neighbor and nearest-hyperrectangle algorithms. Machine Learning, 19(1): 5-28.

Wilson, D.R., and Martinez, T.R. (1996). Value difference metrics for continuously valued attributes. Proceedings of the International Conference on Artificial Intelligence, Expert Systems and Neural Networks, p. 11-14.

Xiaoli, Q.A. (1999). Case-based reasoning system for bearing design, [MSc. thesis]. Faculty of Computer Science, Drexel University, Philadelphia, PA.