Classification model with subspace data-dependent balls

Nattapon Klakhaeng, Thanapat Kangkachit, Thanawin Rakthanmanon and Kitsana Waiyamai
Data Analysis and Knowledge Discovery Lab
Department of Computer Engineering, Faculty of Engineering, Kasetsart University.
Bangkok, Thailand
g5414550051@ku.ac.th, g5185041@ku.ac.th, thanawin.r@ku.ac.th and fengknw@ku.ac.th,

Abstract—Data-Dependent Ball (DDB) is a pre-processing algorithm that transforms quantitative into binary data by mapping them into a set of balls. In datasets with large number of dimensions, data-dependent balls are less significant due to the distance calculation in the mapping process. To reduce number of ball dimensions, this paper proposes a method for subspace data-dependent balls (SDDB) generation. SDDB starts by ranking features using information gain, and then eliminating input features based on ratio $r$. Subspace data-dependent balls are then created and filtered out with respect to their size and purity. Finally, a C4.5 decision tree classification model is constructed using subspace data-dependent balls as features. Experimental results from 8 UCI datasets show that the accuracy from a combination of SDDB and C4.5 is better than the combination of DDB and C4.5 in terms of accuracy.

Keywords—subspace data-dependent balls, classification, feature selections

I. INTRODUCTION

Discretization[1] is a technique that transforms a quantitative attribute into a small number of intervals. It has been recognized that discretization is helpful in improving accuracy of classification tasks of data mining. Discretization can be categorized into 2 categories which are unsupervised and supervised methods. Unlike unsupervised discretization methods which do not use class label during the discretization process, supervised discretization methods such as MDL[2], CAIM[3] and CACC[4] use class label to help the discretization process. These methods use information gain to recursively determine the best cut points. In terms of classification accuracy, supervised methods are much better than unsupervised methods. However, most of the supervised methods use only one attribute for discretization that generates non-overlapped intervals. It must be noticed that, real-world data can contain quantitative attributes that can be mapped into clusters of overlapped intervals, as shown in Fig. 1.

A number of methods that use multi-attributes to transform quantitative into categorical attributes have been proposed [5-10]. They transform a quantitative into categorical attribute by using a clustering algorithm and then use cluster-based interval to map data into clusters. Without considering class-label, these methods generate impure clusters that might reduce performance of classification accuracy. In [10, 11], recursive clustering techniques have been proposed to obtain clusters with high purity. However, these techniques generate too many clusters which make the classification more complex and hard to understand.

Fig. 1. Transformation of quantitative attribute into clusters.

Data-Dependent Ball (DDB) [12] is an alternative algorithm that uses multi-attribute to transform quantitative attributes. Main advantage of this method is its comprehensibility. Each ball has a precise boundary that can be converted into a number of intervals. A data-dependent ball can be considered as a binary feature, where distance between ball center and example can be calculated to determine all the examples that the ball are covered. However, DDB creates too many balls and the distance between two objects in high-dimensional space is meaningless (known as “curse of dimensionality”). To reduce number of dimensions, [13] used kernel technique for data transformation. Such method generates dependent balls that are difficult to explain their meaning.

In this paper, we propose a classification method based on subspace data-dependent balls. To improve classification accuracy, we develop a method named SDDB to generate subspace data-dependent balls. SDDB starts by ranking features using information gain, and then eliminate input features based on ratio $r$. Subspace data-dependent balls are then created and filtered out with respect to their size and purity.
To evaluate performance of SDDB, a C4.5 decision tree is used as base classifier. Various pre-processing techniques are used to generate decision trees, and comparison based on accuracy of these decision trees is made. Experimental results on 8 UCI datasets show that the accuracy from a combination of C4.5 with SDDB yields better accuracy on most datasets.

To summarize, the contributions of our work are as follows

- Find subspace features for each class and assign the subspace to each data-dependent ball.
- Reduce number of balls by filter out small-size and impurity balls.

The rest of this paper is organized as follows. In Section 2, we review data-dependent ball and its algorithm. Then, our proposed algorithm SDDB is described in section 3. In section 4, we show the experimental results and conclude the paper.

II. CLASSIFICATION WITH DATA-DEPENDENT BALLS

A. Data-dependent ball

In this section, we briefly present a data-dependent ball [12, 15]. A data-dependent ball is a function that transforms quantitative into binary data. A data-dependent ball \( b \) can be used as a binary feature which indicates whether the examples are located inside or outside \( b \). Given a training example \( x \) and a ball \( b \) with center \( c \) and radius \( p \), the function \( h \) to determine whether \( x \) is within \( b \) can be formulated as

\[
h_{c,p}(x) = \begin{cases} 
1 & \text{IF } d(c,x) < p \\
0 & \text{otherwise}
\end{cases}
\]

Where \( d(c,x) \) is the distance between center \( c \) and example \( x \). Here, a traditional Euclidean distance is used for \( d \) (any distance function can be used). To avoid computational complexities, a set of center balls must be chosen from the training examples. Moreover, the set of relevant radius values are given based on distance between a center and another border example. Same as in Decision List Machine (DLM) [15], each ball is given a class label with highest usefulness \( U \) (given by (2)).

For example, let consider 3 training examples \( \{ x_1, x_2, x_3 \} \) in Fig.2. Each of the six balls is created by assigning each example as its center.

<table>
<thead>
<tr>
<th>Example</th>
<th>Data-Dependent Balls</th>
<th>Class-label</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_1 )</td>
<td>( b_{1,2} ), ( b_{1,3} )</td>
<td>P</td>
</tr>
<tr>
<td>( x_2 )</td>
<td>( b_{1,1} ), ( b_{2,2} ), ( b_{3,1} )</td>
<td>P</td>
</tr>
<tr>
<td>( x_3 )</td>
<td>( b_{2,1} ), ( b_{2,3} ), ( b_{3,2} )</td>
<td>N</td>
</tr>
</tbody>
</table>

Let \( b_{c,e} \) be a ball with center \( x_e \) and \( x_i \) be its border. Let \( x_a \) be a training example located inside the ball \( b_{c,e} \), and then the mapping between \( x_a \) and feature \( b_{c,e} \) is equal to 1. Otherwise, it is equal to 0. For example, consider training example \( x_3 \) in

Table I. \( x_3 \) is covered by a set of balls \( \{ b_{1,3}, b_{2,2}, b_{3,2} \} \). Thus, only these feature balls are equal to 1 for \( x_3 \).

B. Decision List Machine

In this section, we briefly present a Decision List Machine (DLM) classification algorithm [14]. DLM uses data-dependent balls as classification features. It is a greedy algorithm that produces smallest error on the training dataset. Recursively, DLM finds balls with highest usefulness \( U_c \) defined as follows:

\[
U_c = |P| - p_c \cdot |N|
\]

where \( c \) is a class label, \( p_c \) is a penalty for class \( c \), \(|P|\) and \(|N|\) is the number of positive and negative examples for class \( c \) which is located inside a ball, respectively. However, penalty value is quite difficult to set; too high penalty will cause a negative value of usefulness. This might decrease classification accuracy. Instead, we use information gain to rank the data-dependent balls.

III. METHODOLOGY

In this section, our method named Subspace Data-Dependent Ball (SDDB) is described. SDDB consists of 3 parts. First part (A) is to find subspace features for each class. It is composed of 3 steps; split data, rank features based on information gain and eliminate input features. Second part (B) is to create subspace data-dependent balls. It is composed of 3 steps; generate all the possible subspace ball, map data into balls, and filter out useless balls. Third part (C) is to build classification with data-dependent balls, and is composed of 2 steps; rank balls based on information gain and compute class-label of the selected balls. Overview of SSDB is shown in Fig. 3.
A. Find Subspace Features

At this step, SDDB aims at finding the best percentage subspace features for each class. SDDB starts by splitting data according to their class label. Dummy technique [15] with class-label features is used to transform k class-label features into k binary features. An example of dummy technique is shown in Fig. 4. The input data with class label are first transformed into the k class-label features. Then it is splitted into 3 tables according to the number of class labels. Finally, for each table, features are ranked based on information gain [16] which is the change of information entropy as defined as:

\[ \text{InfoGain}(C, A) = H(C) - H(C \mid A) \]  

where \( H \) denotes the Shannon entropy, \( C \) is a set of class-labels (true or false) and \( A \) is a given feature. The last step of this part is to select top subspace features according to ratio \( r \). The input features are ranked based on information gain (3) and use ratio \( r \) to select \( k \) subspaces as follows:

\[ \text{SelectFeature}(r, S) = (1 - r) \cdot NF(S) \]  

where \( NF(S) \) is the number of features in the data \( S \).

B. Create Subspace Data-Dependent Balls

In this step, data-dependent balls are created by taking into account of the selected subspace features of each class. To determine whether an example \( x \) is located within the ball \( b \) with center \( c \) having class-label \( i \), distance function \( d \) is modified by considering only a set of selected subspace features \( S \) of class-label \( i \) as formulated as:

\[ h_{c,p}(x) = \begin{cases} 1 & F \ d(c, x, S) < p \\ 0 & \text{otherwise} \end{cases} \]  

Unfortunately, too large numbers of balls are created. This is a time and memory consuming process. SDDB removes useless balls by filtering out balls with size less than \( \text{BallSize} \) threshold and with purity less than \( \text{BallPurity} \) threshold.

C. Build Classifier Model

In this step, C4.5 decision tree is used as classification model. The set of subspace dependent balls is used as binary attributes. Therefore only binary attributes are used in C4.5. When the stopping criterion is satisfied, decision node is labeled with a majority class of examples within data-dependent ball.

Fig. 5, shows classification model created by SDDB and DLM. It also shows that DLM rules can be generated from SDDB decision tree. Notice that DLM is a classification algorithm that selects a set of balls to construct classification model by ranking them based on usefulness. However in SDDB, balls are ranked based on information gain.
In improving classification accuracy. Parameters are set as follows. Ratio r varies from 0.0 to 0.5. Experiment results show that ratio r = 0.4 is the best ratio to select the subspace features. Notice that classification accuracy is improved for six datasets.

IV. EXPERIMENTAL RESULTS

To evaluate performance of SDBD, we compare accuracy of C4.5 that constructed from SDBD and from other preprocessing methods. We design two sets of experiments. The first experiment is to study performance of subspace data-dependent balls in improving classification accuracy. The second experiment is to compare classification accuracy with other pre-processing methods.

Experiments are conducted on 8 UCI datasets as illustrated in TABLE II. 3 of 8 datasets are imbalance datasets i.e. glass, haberman and ionosphere. Datasets are normalized with z-transform. Classification model are created with decision tree C4.5 (use J48 with default parameter. in Weka as C4.5)

A. Performance of subspace data-dependent balls in improving classification accuracy

In this experiment, we want to study benefits of SDBD in improving classification accuracy. Parameters are set as follows. Ratio r varies from 0.0 to 0.5, purity=0.99 and size=0.01. The performance of SDBD is measured using 10 folds cross validation.

To find out the best accuracy, SDBD needs to find out best ratio r of each dataset. And this will depend on characteristics of each dataset. Table IV shows the best values of ration r, and best subspace obtained by SDBD. Table V shows the accuracy obtained when using the best subspace.

B. Compare classification accuracy of SDBD with other pre-processing methods

This experiment is to compare classification accuracy of SDBD with other pre-processing methods. The performance of prediction models is measured using 10 folds cross validation. The base-line performance is based on the following algorithms: C4.5 [17], MDL [2] with C4.5, data-dependent ball with C4.5 (DDB), subspace decision clustering tree (SDCC) [11] and best subspace data-dependent ball with C4.5 (BestSDBD).

According to Table V, the performance of C4.5 with DDB is better than C4.5. C4.5 with SDBD is the best performance in term of average accuracy from 8 datasets. DDB with C4.5 uses decision boundary with balls, but C4.5 uses decision boundary with interval of data. With SDBD, distances between subspace data-dependent balls with data examples are much more significant.

In order to understand decision boundary obtained from different methods, we generate synthetic dataset with clear decision boundary as shown in Fig 6a. The dataset contains 2 attributes and 2 classes. C4.5 uses interval of data to make decision boundary that is orthogonal. Fig 6b, shows decision boundary produced by C4.5. This has impact on classification accuracy when decision boundary is not orthogonal. However,
combination of DDB and C4.5 produces a real decision boundary based on the dependent balls. Fig 6c, shows decision boundary obtained from combination of DDB and C4.5.

### Table IV. Number of Eliminated Features with Ratio $r$ That Give Best Accuracy for Decision Tree

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Full-Space</th>
<th>Best-$r$</th>
<th>Eliminate Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sonar</td>
<td>61</td>
<td>0.03</td>
<td>1</td>
</tr>
<tr>
<td>ionosphere</td>
<td>35</td>
<td>0.67</td>
<td>2</td>
</tr>
<tr>
<td>haberman</td>
<td>4</td>
<td>0.33</td>
<td>3</td>
</tr>
<tr>
<td>glass</td>
<td>10</td>
<td>0.03</td>
<td>2</td>
</tr>
<tr>
<td>iris</td>
<td>4</td>
<td>0.33</td>
<td>1</td>
</tr>
<tr>
<td>wine</td>
<td>12</td>
<td>0.55</td>
<td>6</td>
</tr>
<tr>
<td>vehicle</td>
<td>19</td>
<td>0.39</td>
<td>7</td>
</tr>
<tr>
<td>ecoli</td>
<td>8</td>
<td>0.13</td>
<td>1</td>
</tr>
</tbody>
</table>

**Table V. Comparison on Average Accuracies between (SDDB + C4.5) and Related Methods**

<table>
<thead>
<tr>
<th>Datasets</th>
<th>C4.5</th>
<th>C4.5+MDL</th>
<th>SDCC</th>
<th>C4.5+DDB</th>
<th>C4.5+Best SDDB</th>
</tr>
</thead>
<tbody>
<tr>
<td>ionosphere</td>
<td>89.46</td>
<td>89.17</td>
<td>72.36</td>
<td>94.02</td>
<td>94.02</td>
</tr>
<tr>
<td>haberman</td>
<td>72.88</td>
<td>71.90</td>
<td>67.97</td>
<td>72.55</td>
<td>72.88</td>
</tr>
<tr>
<td>glass</td>
<td>64.02</td>
<td>73.83</td>
<td>53.27</td>
<td>68.22</td>
<td>69.63</td>
</tr>
<tr>
<td>sonar</td>
<td>69.23</td>
<td>79.81</td>
<td>75.48</td>
<td>84.13</td>
<td>84.62</td>
</tr>
<tr>
<td>iris</td>
<td>96.00</td>
<td>94.00</td>
<td>89.33</td>
<td>91.33</td>
<td>96.00</td>
</tr>
<tr>
<td>wine</td>
<td>93.26</td>
<td>93.82</td>
<td>73.03</td>
<td>93.82</td>
<td>97.19</td>
</tr>
<tr>
<td>vehicle</td>
<td>72.93</td>
<td>71.99</td>
<td>31.91</td>
<td>62.53</td>
<td>70.45</td>
</tr>
<tr>
<td>ecoli</td>
<td>72.93</td>
<td>81.55</td>
<td>58.33</td>
<td>75.89</td>
<td>76.79</td>
</tr>
</tbody>
</table>

| Imbalance | 75.45 | 78.30 | 64.54 | 78.26 | 78.44 |
| Balance | 80.87 | 84.23 | 65.62 | 81.54 | 85.01 |
| Average | 78.84 | 82.01 | 65.21 | 80.31 | 82.70 |

V. Conclusion

SDDB is a pre-processing method to find subspace data-dependent balls. Experimental results show that SDDB give better accuracy than data-dependent balls. This can be explained by the fact that real-world data may contain noise in their attributes. Thus some (not all) attributes are required for data classification task. Those attributes may also contain noise. The proposed method is not yet suitable for imbalance datasets. The reason is that small number of balls is generated for the minority class. In addition data-dependent ball has some disadvantages by generating too large number of attributes with similar value.

Our future work consists in reducing more similar attributes, improving the process of finding subspace feature for data-depend balls. Key research direction is also to provide a mechanism to extract more knowledge from the subspace data-dependent balls.

![Decision boundary comparison between DDB and other pre-process methods](image)
REFERENCES


