Improving the Performance of Minor Class in Decision Tree Using Duplicating Instances

HYONTAI SUG
Division of Computer and Information Engineering
Dongseo University
Busan, 617-716
REPUBLIC OF KOREA
hyontai@yahoo.com http://kowon.dongseo.ac.kr/~sht

Abstract: Because decision trees are built to cover all training instances with minimal errors, it is true that the instances that belong to minor classes are treated less importantly in classification. As a result, the classification accuracy for minor classes is usually poorer than that of major classes. But we hope that the classification is also good for the minor classes. This paper suggests to use over-sampling for minor classes to generate more accurate trees for minor classes, and use decision trees with conventional sampling method as well as decision trees with the over sampling method together for better classification. Experiments with a representative decision tree algorithm, C4.5, shows very promising results.

Key-Words: decision trees, biased sampling, class imbalance.

1 Introduction
There are two good points of decision trees. The first good point is in their structures that are relatively easy to understand, and the second good point is scalability that enables us to deal with relatively large data sets easily. But even though their good points in structure and scalability, there is some weak point due to the fact that their branching criteria give higher priority for major classes based on greedy algorithms. Because of the greedy property of decision tree generation algorithms, as a decision tree is being built, each subtree in the decision tree becomes to have less training examples as the result of branching, and the instances of major class are considered more importantly than the instances minor class. Moreover, the reliability of lower branches becomes worse than upper branches due to the smaller size of training examples. So, the classification for minor classes that occur rarely in the target data set is suffered much more than that of major classes.

If target databases for data mining are very large, we may resort to sampling. Some notable fact in sampling is that the trained knowledge model based on the samples is likely dependent on the samples. It is known that decision tree algorithms are more dependent upon training data sets, because decision tree algorithms divide the data sets decisively, while some other data mining tools like artificial neural networks supply all training instances simultaneously to all of their networks [1].

In order to overcome the problem of disdaining minority classes in decision tree generation algorithms, we need some technique to treat the minor classes better so that the minor classes are treated more heavily in the decision tree algorithms. In this paper we suggest some progressive way in sampling that allows to consider minor classes more importantly for decision trees.

In section 2, we provide the related work to our research, and in sections 3 we present our method. Experiments were run to see the effect of the method in section 4. Finally section 5 provides some conclusions.

2 Related Work
Decision tree algorithms use some greedy search methods to split branches so that generated decision trees may not be optimal. There have been a lot of efforts to build better decision trees and splitting measure is a major concern. For example, C4.5 algorithm that is often referred in literature [2] uses an entropy-based measure, and the measure prefers the most certain split among possible splits from candidate features. So major classes are preferred, because there are more instances of major classes in the data set, and usually more certain in splitting.
Scalability in decision trees was also a good issue for research. Some representatives are like SLIQ, SPRINT, PUBLIC, and SURPASS. SLIQ [3] saves some computing time when the data set consists of many continuous attributes by using a pre-sorting technique in tree-growth phase, and SPRINT [4] is an improved version of SLIQ to solve the scalability problem by building trees with parallel processing algorithm. PUBLIC [5] tries to save some computing time by integrating the tasks of pruning and generating branches together. SURPASS [6] solved the problem of large data set size by bringing the portion of data set into main memory that are needed to grow branches at the moment. However, these methods may generate large decision trees so that the problem of comprehensibility and neglecting minor classes still may occur.

Because training of decision trees is a kind of induction, and the data is fragmented in the training process, the performance of trained decision tree is dependent on the training data set a lot. So, we can infer that the resulting decision trees may be dependent on the composition of data in the data set. SMOTE method [7] used synthetic data generation method for minor classes, and showed that it is effective for decision trees. In [8] the authors showed that class imbalance has different effect in neural networks for medical domain data.

3 The Method

Because decision tree algorithms do not give high priority to minor classes in splitting branches, it is highly possible that instances of minor classes are treated in the lower part of the tree, and this treatment may increase misclassification rate for minor classes. So we want decision tree algorithms to treat the instances of minor classes more importantly. In order to do this, we increase the number of instances of minor classes by duplication. Moreover, in order to decide a good duplication rate, we increase the percentage of duplication progressively. The following is a brief description of the procedure of the method.

INPUT: a data set for data mining,
K: the percentage of over-sampling,
L: threshold of change,
X: the number of times to do sampling.
Y: sample size.

OUTPUT: better decision trees with respect to minor class.

Begin
Do random sampling of size of X, Y times.
For each sample data set Do
    Generate a decision tree for original sample data;
    Make confusion matrix with test data;
    n:=number_of_false_classification_in_minor_class;
    Do while L < | n-m |
        n := m;
        Duplicate the instances of minor class by K%;
        /* increase K% more*/
        Generate a decision tree;
        Make a confusion matrix using test data;
        m:=number_of_false_classification_in_minor_class;
    End while;
End Do;
End.

In the algorithm we duplicate the instances in minor class until the change in false classification for minor class reaches to some predefined limit, L. In the following experiment given L value is 50. We can also set K percentage of duplication for each iteration in the while loop. In the following experiment given K value is 100%, and 4 and 16,000 for X and Y respectively.

4 Experimentation

Experiments were run using a database in UCI machine learning repository [9] called 'adult' [10] to see the effect of the method. The number of instances is 48,842. Class probabilities for label <=50K and >50K are 76.07% and 23.93% respectively, and class >50K is the minor class. The database was selected because it is relatively large and contains lots of values. The total number of attributes is 14, and among them six are continuous attributes and eight are nominal attributes.

C4.5 was used to generate decision trees for five sample sets of size 16,000. Remaining data are used for test. The following Table 1 to 4 show accuracy and confusion matrix in minor class over-sampling for four different sample sets of size 16,000.
Table 1. Confusion matrix of decision tree by C4.5 with various percentages of over-sampling for minor class for sample set 1

<table>
<thead>
<tr>
<th>Over-sampling Ratio: accuracy</th>
<th>True ‘&gt;50K’</th>
<th>False ‘&gt;50K’</th>
<th>False ‘&lt;=50K’</th>
<th>True ‘&lt;=50K’</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original: 85.67%</td>
<td>4696</td>
<td>3157</td>
<td>1550</td>
<td>23441</td>
</tr>
<tr>
<td>200%: 83.49%</td>
<td>5660</td>
<td>2191</td>
<td>3230</td>
<td>21761</td>
</tr>
<tr>
<td>300%: 81.28%</td>
<td>6075</td>
<td>1776</td>
<td>4373</td>
<td>20618</td>
</tr>
<tr>
<td>400%: 80.46%</td>
<td>6077</td>
<td>1774</td>
<td>4642</td>
<td>20349</td>
</tr>
</tbody>
</table>

If we look at table 1, the difference of false ‘>50K’ between 300% and 400% minor class over-sampling is only two, so we stop further over-sampling.

Table 2. Confusion matrix of decision tree by C4.5 with various percentages of over-sampling for minor class for sample set 2

<table>
<thead>
<tr>
<th>Over-sampling Ratio: accuracy</th>
<th>True ‘&gt;50K’</th>
<th>False ‘&gt;50K’</th>
<th>False ‘&lt;=50K’</th>
<th>True ‘&lt;=50K’</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original: 85.51%</td>
<td>4462</td>
<td>3398</td>
<td>1362</td>
<td>23620</td>
</tr>
<tr>
<td>200%: 83.04%</td>
<td>5744</td>
<td>2116</td>
<td>3455</td>
<td>21527</td>
</tr>
<tr>
<td>300%: 81.27%</td>
<td>6045</td>
<td>1815</td>
<td>4336</td>
<td>20646</td>
</tr>
<tr>
<td>400%: 80.67%</td>
<td>6081</td>
<td>1773</td>
<td>4575</td>
<td>20407</td>
</tr>
</tbody>
</table>

If we look at table 2, the difference of false ‘>50K’ between 300% and 400% minor class over-sampling is only two, so we stop further over-sampling.

Table 3. Confusion matrix of decision tree by C4.5 with various percentages of over-sampling for minor class for sample set 3

<table>
<thead>
<tr>
<th>Over-sampling Ratio: accuracy</th>
<th>True ‘&gt;50K’</th>
<th>False ‘&gt;50K’</th>
<th>False ‘&lt;=50K’</th>
<th>True ‘&lt;=50K’</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original: 85.63%</td>
<td>4874</td>
<td>2976</td>
<td>1745</td>
<td>23249</td>
</tr>
<tr>
<td>200%: 83.73%</td>
<td>5679</td>
<td>2171</td>
<td>3173</td>
<td>21819</td>
</tr>
<tr>
<td>300%: 81.95%</td>
<td>5904</td>
<td>1946</td>
<td>4575</td>
<td>20407</td>
</tr>
</tbody>
</table>

If we look at table 3, the difference of false ‘>50K’ between 300% and 400% minor class over-sampling is only two, so we stop further over-sampling.

Table 4. Confusion matrix of decision tree by C4.5 with various percentages of over-sampling for minor class for sample set 4

<table>
<thead>
<tr>
<th>Over-sampling Ratio: accuracy</th>
<th>True ‘&gt;50K’</th>
<th>False ‘&gt;50K’</th>
<th>False ‘&lt;=50K’</th>
<th>True ‘&lt;=50K’</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original: 85.62%</td>
<td>4600</td>
<td>3195</td>
<td>1528</td>
<td>23519</td>
</tr>
<tr>
<td>200%: 83.65%</td>
<td>5611</td>
<td>2184</td>
<td>3185</td>
<td>21862</td>
</tr>
<tr>
<td>300%: 81.93%</td>
<td>5971</td>
<td>1824</td>
<td>4112</td>
<td>20935</td>
</tr>
<tr>
<td>400%: 80.91%</td>
<td>6091</td>
<td>1704</td>
<td>4647</td>
<td>20400</td>
</tr>
</tbody>
</table>

If we look at table 4, the difference of false ‘>50K’ between 400% and 500% minor class over-sampling is only two, so we stop further over-sampling.

Let’s think of how we use the trees, and assume that DT1 is a decision tree generated from original sample data set, and DT2 is the best decision tree with respect to the number of false classification for minor class from over-sampling. According to the result of experiment DT1 has good accuracy for the major class. On the other hand, DT2 is good for the minor class. But the confidence of each terminal node in DT2 is originated from the over-sampled data set, so that it is somewhat exaggerated. So, we need to modify the confidence of each terminal node of DT2 with test data set.

In order to classify class-unknown instances, we try to classify them using both trees. If the two decision trees classify an instance as it belongs to the same class, we decide it is in the class. If it is classified differently, we trace the branches of the two trees, and select a class that has higher confidence. In the above experiment, we can have eight decision trees and we
may use the decision tree in table 5 to break a tie, if we use voting.

<table>
<thead>
<tr>
<th>Over-sampling Ratio: accuracy</th>
<th>True ‘$&gt;$50K’</th>
<th>False ‘$&gt;$50K’</th>
<th>False ‘$&lt;=$50K’</th>
<th>True ‘$&lt;=$50K’</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original: 85.53%</td>
<td>4563</td>
<td>3343</td>
<td>1408</td>
<td>23528</td>
</tr>
</tbody>
</table>

Table 5. Decision tree by C4.5 with various percentages of over-sampling for minor class for sample set 5

5 Conclusions
Decision trees have been considered one of good data mining tools with respect to understandability and scalability. Even though the good points, there is some weak point due to the fact that their branching criteria give higher priority for major classes based on greedy algorithms. So, the classification for minor classes that occur rarely in the target data set is suffered more than that of major classes.

If target databases for data mining are very large, we may resort to sampling. Some noticable fact in sampling is that the trained decision trees are highly dependent on the samples.

In order to overcome the problem of disdaining minority classes in decision tree generation algorithms, we resort to a technique of progressive over-sampling for minor classes with duplication. The resulting decision trees can be used with voting method to predict classes for future and unseen instances. Experiments with a real world data set and a decision tree algorithms, C4.5, showed very promising result.

References: