International Journal of Pattern Recognition
and Artificial Intelligence
Vol. 29, No. 4 (2015) 1551004 (18 pages)

DOI: 10.1142/S0218001415510040

World Scientific
www.worldscientific.com

A Novel Minority Cloning Technique for Cost-Sensitive Learning

Liangxiao Jiang* and Chen Qiu†
Department of Computer Science, China University of Geosciences
Wuhan, Hubei 430074, P. R. China
**ljiang@cug.edu.cn*
†*qiuchen1114@gmail.com*

Chaoqun Li
Department of Mathematics, China University of Geosciences
Wuhan, Hubei 430074, P. R. China
chqli@cug.edu.cn

Received 25 October 2014
Accepted 13 February 2015
Published 27 April 2015

In many real-world applications, it is often the case that the class distribution of instances is imbalanced and the costs of misclassification are different. Thus, the class-imbalanced cost-sensitive learning has attracted much attention from researchers. Sampling is one of the widely used techniques in dealing with the class-imbalance problem, which alters the class distribution of instances so that the minority class is well represented in the training data. In this paper, we propose a novel Minority Cloning Technique (MCT) for class-imbalanced cost-sensitive learning. MCT alters the class distribution of training data by cloning each minority class instance according to the similarity between it and the mode of the minority class. The experimental results on a large number of UCI datasets show that MCT performs much better than Minority Oversampling with Replacement Technique (MORT) and Synthetic Minority Oversampling TEchnique (SMOTE) in terms of the total misclassification costs of the built classifiers.

Keywords: Instance cloning; similarity; cost-sensitive learning; misclassification cost.

1. Introduction

In many real-world applications, the class distribution of instances is typically imbalanced,[21] and the degree of imbalance (the ratio of the size of the majority class to the size of the minority class) varies from one application to another and a correct classification of the minority class often has greater value than that of the majority class in these applications.[3,13,26]

*Corresponding author.

Because the instances that belong to the minority class occur infrequently, the models that describe the minority class (the rare class) tend to be highly specialized. Therefore, detecting the instances that belong to the rare class is very difficult and many of the existing classifiers are not qualified for this task.

The basic problem setting of machine learning is that, given a set of training instances drawn from a fixed underlying distribution, a learner induces a model to best approximate the distribution through best fitting the training data. Most machine learning research is concentrated on the effective learning algorithms inducing an accurate model to fit the training data, while relatively less effort has been made on how to convert the naturally occurring training data into a form suitable to the learning algorithm.

Although some research suggests that using the natural data yields the best classification accuracy,[2] it is not true in general. First, the natural training data may not obey the underlying distribution in reality. For example, the training instances may come from different sources and may be collected independently.[11] Second, the distribution of the training data is not suitable for learning. For example, the class distribution may be highly imbalanced. In addition, when the misclassification costs[12,24,35,36] are taken into account, it has been shown that the balanced class distribution performs best, instead of the natural class distribution. Therefore, converting the natural training data into a form suitable to the learning algorithm is often necessary.

Manipulating the natural training data is not new to machine learning community. Here, we restrict our discussion to the classification problem, in which a classifier is induced from a set of labeled training instances, each of which is represented by a vector of attribute values and a class label. Then, the training data have actually two dimensions: the horizontal dimension corresponding to the attributes, and the vertical dimension corresponding to the instances.

Feature selection,[8,16,19] which refers to choosing a subset of attributes that best fits some characteristics of the problem, has received a considerable attention for decades. Feature selection is actually the manipulation on the horizontal dimension of the training data. The manipulation on the vertical dimension corresponds to altering the distribution of training instances. A considerable amount of work has been done on altering the class distribution of the training data. When the natural training data is highly skewed, its class distribution should be changed in order to obtain an accurate classifier. Sampling is often used for this purpose.

In this paper, we propose a novel Minority Cloning Technique (MCT) to alter the class distribution of training data. MCT alters the class distribution of training data by cloning each minority class instance according to the similarity between it and the mode of the minority class. The experimental results on a large number of UCI datasets show that MCT performs much better than Minority Oversampling with Replacement Technique (MORT)[14] and Synthetic Minority Oversampling Technique (SMOTE)[4] in terms of the total misclassification costs.

The rest of this paper is organized as follows. We start by a brief introduction of some related work. Then we propose MCT and report on our experiments, which is followed by the discussion and conclusion.

2. Related Work

The main idea of sampling is just to alter the class distribution of instances so that the minority class is well represented in the training data. Therefore, sampling is widely used for handling the class-imbalance problem. Weiss and Provost[36] empirically study the effect of the class distribution on decision trees' performance. Their experiments show that using the best class distribution yields a significant reduction in error rate compared with using the natural class distribution. They propose a progressive sampling algorithm for selecting training instances that starts with a small training set and progressively adds training instances using a geometric sampling schedule. Cross-validation is used to determine the proportion of minority-class and majority-class instances added in each iteration. Some of the other available approaches for sampling include undersampling, oversampling,[14] hybrid sampling,[17] uncertainty sampling,[23] SMOTE,[4] and Majority Weighted Minority Oversampling TEchnique (MWMOTE).[1]

Undersampling alters the class distribution by removing some instances from the majority class. A potential problem with undersampling is that some useful instances may be removed from the majority class and thus results in a less than optimal classifier. In contrast, oversampling[14] alters the class distribution by replicating some instances in the minority class until its size is equal to that of the majority class. Oversampling does not add any new instances into the training data but scales up the training time of classifiers. Hybridsampling[17] is a hybrid of both under-sampling and oversampling. For detail, hybrid sampling uses a combination of undersampling the majority class and oversampling the minority class to achieve an uniform class distribution. Uncertainty sampling[23] was designed to select informative instances to construct binary classifiers by adopting the uncertainty notion underlying the Query By Committee (QBC) approach, instead of by generating a committee of hypotheses to estimate uncertainty. The well-known SMOTE[4] oversamples the minority class by creating synthetic minority class instances. After SMOTE, SMOTEBoost[5] is proposed by combining the SMOTE algorithm and the boosting procedure. Recently, a new method, called MWMOTE, is presented.[1] MWMOTE first identifies the hard-to-learn informative minority class instances and assigns them weights according to their Euclidean distance from the nearest majority class instances, and then generates the synthetic instances from the weighted informative minority class instances using a clustering approach.

3. The Proposed Technique

In this section, we propose a novel MCT to alter the class distribution of training data. Random oversampling technique samples each minority class training instance

with same probability, that is, each minority class instance is treated equally. However, many researchers have pointed that some stored instances are more reliable than others.[7,37,34] So, to minority class instances, we would like these trustworthy instances to have more drawing power than others. Wilson[37] presented the Edited Nearest neighbor (ENN) technique to improve the performance of Nearest Neighbor algorithm. ENN removes each instance in training data if the instance is misclassified using the vote of its k-nearest neighbors (KNN). Since noisy instances and border instances are more likely to be misclassified by their KNN, ENN edits out noisy instances and close border instances, and at the same time, retains all internal instances. Wang *et al.*[34] developed a simple adaptive distance measure to improve nearest rule, they called their algorithm adaptive nearest neighbor (ANN). Let $d(x, x_i)$ be the Euclidean distance, where x_i is the ith nearest neighbor of the instance x. ANN defines a locally adaptive distance between x and x_i as $d_{\text{new}}(x, x_i) = d(x, x_i)/r_i$, where r_i is defined to be the radius of the largest sphere centered on x_i that excludes all training instances of other classes. Obviously spheres associated with internal instances will have relatively larger radii than those associated with border instances. As a result, the instances inside the class may be more likely to be the nearest neighbors of a query instance than the border instances. The idea underlying ANN is that training instances with larger spheres are closer to the class centers and their class labels are more reliable.

According to the improved performance of ENN and ANN, it is easy to see that the instances inside the class are more trustworthy than the border instances. So when we try to clone the minority class instances, we also should let the internal instances contribute more than the border instances. How can we do that? In this paper, we present a very simple technique which alters the class distribution of training data by cloning each minority class instance according to the similarity between it and the mode of the minority class. The higher the similarity, the closer the instance is to the center of the class. These internal instances will be cloned more than the border instances. The detailed explanation is given as follows.

The same as previous techniques, we also restrict our discussion to binary classification problems, and the minority class is denoted as the positive class (+), while the majority class is denoted as the negative class (−). Before we describe our proposed MCT algorithm, let us rewrite three important definitions[15,18] as follows. In our current version, just for simplicity, we assume all instances only have nominal attribute values and correspond to points in an m-dimensional discrete space.

Definition 1. *The mode of an attribute values set is the attribute value which has the highest frequency in the attribute values.*

Definition 2. *The mode of an instances set is the instance whose each attribute value is the mode of an attribute values set. In most cases, the mode of an instances set is a constructed virtual instance instead of an actual instance in the instances set.*

Definition 3. *Let x and y be two instances, then the similarity between them is*:

$$s(x, y) = \sum_{i=1}^{m} \delta(a_i(x), a_i(y)), \tag{1}$$

where m is the number of attributes, $\delta(a_i(x), a_i(y))$ is one if $a_i(x) = a_i(y)$ and zero otherwise.

Given a training instances set D, MCT firstly copies D as D_{new} and partitions D into two disjoint subsets according to their class labels. Let D_+ be the positive class (the minority class) and D_- be the negative class (the majority class). Secondly, MCT calculates the mode x of the minority class D_+. Thirdly, MCT measures the similarity $s(x, y)$ between the mode x and each minority class instance y from D_+ and adds $s(x, y)$ clones of y into D_{new}. At last, MCT builds a base classifier C on D_{new} and returns the built classifier. Now, the whole learning algorithm of MCT can be depicted as:

Algorithm: MCT (D, C)
Input: a training set D and a base classifier C
Output: the built cost-sensitive classifier

(1) Copies D as D_{new}
(2) Partitions D into two disjoint subsets according to their class labels
(3) Let D_+ be the minority class
(4) Calculates the mode x of D_+
(5) For each minority class instance y from D_+
  (a) Measures the similarity $s(x, y)$ between the modes x and y
  (b) adds $s(x, y)$ clones of y into D_{new}
(6) Builds a base classifier C on D_{new}
(7) Returns the built cost-sensitive classifier

MCT is different from the MORT.[14] Our MCT clones each minority class instance according to the similarity between it and the mode of the minority class while MORT randomly oversamples the minority class with replacement. At the same time, our MCT is also totally different from the SMOTE.[4] SMOTE oversamples the minority class by creating synthetic minority class instances.

Compared to the used base classifier, our MCT needs some additional time to clone each minority class instance. The time complexities of three key steps (Step 2, Step 4, and Step 5) are $O(n)$, $O(mn_+)$, and $O(mn_+)$, respectively, where n is the number of training instances, m is the number of attributes, and n_+ is the number of the positive instances (the number of the minority class instances). In a word, our MCT takes time complexity of $O(n + 2mn_+)$ to finish the cloning process prior to building the used base classifier.

To the best of our knowledge, cost-sensitive learning is generally categorized into two categories.[25] One is the direct method, which is to directly introduce and utilize

misclassification costs into the learning algorithms. As a result, the learning algorithms are cost-sensitive in themselves. The other category is called cost-sensitive meta-learning method, which converts existing cost-insensitive learning algorithms into cost-sensitive ones by pre-processing the training data or post-processing the output. Cost-sensitive meta-learning method can be further classified into two main subcategories: threshold adjusting and sampling. Thus it can be seen that our proposed MCT is substantially an oversampling method.

4. Experiments

4.1. *Evaluation criteria*

To binary classification problems, four kinds of misclassification costs are needed to provide, which we refer to as C_{TP}, C_{FP}, C_{TN}, and C_{FN}, respectively. C_{TP} and C_{TN} are the costs of true positive (TP) and true negative (TN), as is often the case in cost-sensitive learning, C_{TP} and C_{TN} are set to 0. C_{FN} and C_{FP} are the costs of false negative (FN) and false positive (FP). We always assume that the cost of misclassifying positive class instances is much higher than that of misclassifying negative class instances, so we set $C_{\mathrm{FN}} \gg C_{\mathrm{FP}}$. In this paper, C_{FP} is set to be a unit cost of 1, C_{FN} is assigned different values: 5, 10, and 15, respectively.[12,24,35] In our experiments, the total misclassification costs is used to evaluate the performance of the built cost-sensitive classifiers, which is defined as

$$\mathrm{Costs} = \mathrm{FN} \times C_{\mathrm{FN}} + \mathrm{FP} \times C_{\mathrm{FP}}. \tag{2}$$

4.2. *Experimental settings*

The purpose of these experiments is to validate the effectiveness of the proposed MCT. We run our experiments on 14 UCI datasets.[27] The detailed description of these datasets used in our experiments is shown in Table 1. We select them because

Table 1. Descriptions of UCI datasets used in the experiments.

Dataset	#Instances	#Attributes	#Maj/#Min	Ratio	Missing	Numeric
adult	4884	14	3717/1167	3.2	Y	Y
bank	4521	16	3989/532	7.5	N	Y
breast-cancer	286	9	201/85	2.4	Y	N
breast-w	699	9	458/241	1.9	Y	N
colic	368	22	232/136	1.7	Y	Y
colic.ORIG	368	27	244/124	2.0	Y	Y
credit-g	1000	20	700/300	2.3	N	Y
diabetes	768	8	500/268	1.9	N	Y
hepatitis	155	19	123/32	3.8	Y	Y
ionosphere	351	34	225/126	1.8	N	Y
labor	57	16	37/20	1.9	Y	Y
sick	3772	29	3541/231	15.3	Y	Y
spect	267	22	212/55	3.9	N	N
vote	435	16	267/168	1.6	Y	N

the number of classes in these datasets is just 2 and their ratios of the size of the majority class to the size of the minority class are above 1.5. In our experiments, numeric variables are discretized using the unsupervised 10-bin discretization implemented in WEKA.[39] Missing values are replaced with the mean values from the available data. Besides, we randomly sampled two large datasets "adult" and "bank" with 10% data size for saving the running time of experiments.

We empirically test our MCT on four of top 10 algorithms in data mining[40]: KNN,[31] naive Bayes (NB),[22] C4.5,[30] and SVM.[20,29,32] We compare MCT with MORT and SMOTE in terms of the total misclassification costs of the built cost-sensitive classifiers. Besides, in order to further validate the effectiveness of our proposed MCT, we compare it with another state-of-the-art cost-sensitive meta-learning method called MetaCost (simply denoted by MC in this paper),[10] which is a general method for making classifiers cost sensitive. We implement our MCT in WEKA 3.7.1 platform[39] and use the existing implementations of MORT, SMOTE, MC, KNN, NB, C4.5, and SVM in WEKA 3.7.1 platform.[39]

In all experiments, the total misclassification costs of a classifier on a dataset was obtained via 10 runs of 10-fold cross-validation. Runs with the various classifiers were carried out on the same training sets and evaluated on the same test sets. Finally, we conducted the corrected paired two-tailed t-test with the $p = 0.05$ significance level[28] and the Friedman test[9] with the corresponding *post-hoc* tests such as Nemenyi and Bergmann tests to compare our proposed MCT with its competitors. Now, we introduce established algorithms and their abbreviations used in our experiments.

(1) KNN: The k-nearest neighbors classifier. (The number of nearest neighbors is set to 5.)
(2) KNN-MCT: MCT with KNN as the base classifier.
(3) KNN-MORT: MORT with KNN as the base classifier. (The percentage of MORT instances to create and the seed used for random sampling are set to 100% and 1, respectively.)
(4) KNN-SMOTE: SMOTE with KNN as the base classifier. (The percentage of SMOTE instances to create, the number of nearest neighbors, and the seed used for random sampling are set to 100%, 5, and 1, respectively.)
(5) KNN-MC: MC with KNN as the base classifier. (The size of each bag, the number of bagging iterations, and the random number seed to be used are set to 100%, 10, and 1, respectively.)
(6) NB: The naive Bayesian classifier.
(7) NB-MCT: MCT with NB as the base classifier.
(8) NB-MORT: MORT with NB as the base classifier. (The percentage of MORT instances to create and the seed used for random sampling are set to 100% and 1, respectively.)
(9) NB-SMOTE: SMOTE with NB as the base classifier. (The percentage of SMOTE instances to create, the number of nearest neighbors, and the seed used for random sampling are set to 100%, 5, and 1, respectively.)

(10) NB-MC: MC with NB as the base classifier. (The size of each bag, the number of bagging iterations, and the random number seed to be used are set to 100%, 10, and 1, respectively.)
(11) C4.5: The decision-tree induction method developed by Quinlan.[30]
(12) C4.5-MCT: MCT with C4.5 as the base classifier.
(13) C4.5-MORT: MORT with C4.5 as the base classifier. (The percentage of MORT instances to create and the seed used for random sampling are set to 100% and 1, respectively.)
(14) C4.5-SMOTE: SMOTE with C4.5 as the base classifier. (The percentage of SMOTE instances to create, the number of nearest neighbors, and the seed used for random sampling are set to 100%, 5, and 1, respectively.)
(15) C4.5-MC: MC with C4.5 as the base classifier. (The size of each bag, the number of bagging iterations, and the random number seed to be used are set to 100%, 10, and 1, respectively.)
(16) SVM: The support vector machines classifier based on John Platt's sequential minimal optimization algorithm (SMO). (The complexity constant C, the epsilon for round-off error, and the tolerance parameter are set to 1.0, $1.0E-12$, and 0.001, respectively. Besides, the polynomial kernel function is used and the training data is normalized.)
(17) SVM-MCT: MCT with SVM as the base classifier.
(18) SVM-MORT: MORT with SVM as the base classifier. (The percentage of MORT instances to create and the seed used for random sampling are set to 100% and 1, respectively.)
(19) SVM-SMOTE: SMOTE with SVM as the base classifier. (The percentage of SMOTE instances to create, the number of nearest neighbors, and the seed used for random sampling are set to 100%, 5, and 1, respectively.)
(20) SVM-MC: MC with SVM as the base classifier. (The size of each bag, the number of bagging iterations, and the random number seed to be used are set to 100%, 10, and 1, respectively.)

4.3. *Experimental results*

Tables 2–5 show the total misclassification costs of each classifier on each dataset when cost ratio is 15. The symbols ∘ and • in the tables respectively denote statistically significant improvement or degradation over its competitors at 95% significance level.[28] The last line shows the win/tie/loss counts of MCT versus its competitors. It is worth noting that, in terms of the total misclassification costs, the larger the number, the worse the performance of the classifier. For example, Table 2 shows that, compared to KNN, the $W/T/L$ value of MCT-KNN is 12/2/0, that means that MCT-KNN significantly outperforms KNN on 12 datasets, ties on two datasets and loses on 0 dataset. Experimental results in Tables 2–5 show that, in terms of the total misclassification costs, KNN-MCT significantly better than KNN, KNN-MORT, and KNN-SMOTE on 12, 10, and 10 datasets, respectively, NB-MCT

Table 2. Comparisons for KNN-MCT versus KNN, KNN-MORT, KNN-SMOTE, and KNN-MC.

Dataset	KNN-MCT	KNN	KNN-MORT	KNN-SMOTE	KNN-MC
adult	264.79 ± 34.27	$819.77 \pm 70.82\circ$	$521.95 \pm 63.04\circ$	$477.82 \pm 59.59\circ$	248.99 ± 32.00
bank	268.37 ± 42.62	$687.83 \pm 37.91\circ$	$562.75 \pm 48.44\circ$	$534.37 \pm 52.60\circ$	$424.16 \pm 67.79\circ$
breast-cancer	35.87 ± 13.47	$95.41 \pm 18.95\circ$	$70.74 \pm 21.40\circ$	$76.29 \pm 22.99\circ$	21.68 ± 5.85 $\bullet$
breast-w	15.35 ± 13.99	$36.50 \pm 21.91\circ$	$23.33 \pm 17.76\circ$	$22.66 \pm 15.61\circ$	$29.47 \pm 18.96\circ$
colic	23.79 ± 8.63	$58.86 \pm 25.33\circ$	$42.13 \pm 19.02\circ$	$37.43 \pm 16.46\circ$	22.66 ± 5.23
colic.ORIG	33.57 ± 13.28	$129.35 \pm 24.50\circ$	$79.20 \pm 24.99\circ$	$59.81 \pm 24.15\circ$	26.93 ± 8.67
credit-g	76.93 ± 19.92	$321.82 \pm 32.16\circ$	$196.17 \pm 34.94\circ$	$155.82 \pm 38.64\circ$	66.67 ± 13.63
diabetes	77.15 ± 24.72	$252.30 \pm 38.14\circ$	$142.73 \pm 33.99\circ$	$134.19 \pm 35.27\circ$	$47.33 \pm 14.12\bullet$
hepatitis	12.51 ± 8.87	$27.49 \pm 13.37\circ$	18.10 ± 12.38	17.38 ± 11.92	10.27 ± 7.99
ionosphere	20.05 ± 14.35	$39.45 \pm 20.69\circ$	$30.60 \pm 18.03\circ$	$31.87 \pm 18.41\circ$	$33.74 \pm 18.30\circ$
labor	2.25 ± 2.80	6.62 ± 8.60	3.18 ± 6.24	3.45 ± 6.06	2.79 ± 3.42
sick	46.05 ± 16.20	$112.22 \pm 31.35\circ$	$72.97 \pm 28.85\circ$	$68.15 \pm 27.78\circ$	$86.32 \pm 30.96\circ$
spect	17.75 ± 8.31	$31.69 \pm 18.18\circ$	20.27 ± 14.47	24.39 ± 15.96	17.33 ± 8.12
vote	11.84 ± 10.26	15.85 ± 13.53	13.96 ± 12.51	15.04 ± 13.13	14.39 ± 11.83
W/T/L	—	12/2/0	10/4/0	10/4/0	4/8/2

significantly better than NB, NB-MORT, and NB-SMOTE on 6, 6, and 8 datasets, respectively, C4.5-MCT significantly better than C4.5, C4.5-MORT, and C4.5-SMOTE on 9, 6, and 7 datasets, respectively, and SVM-MCT significantly better than SVM, SVM-MORT, and SVM-SMOTE with 8, 7, and 8 datasets, respectively. More important, our proposed MCT is competitive with the state-of-the-art cost-sensitive meta-learning method called MC[10] except that C4.5-MCT is notably worse than C4.5-MC on seven datasets.

Tables 6, 8, 10, and 12 respectively show average rankings of the algorithms obtained by applying the Friedman test.[9] With five algorithms and 14 data sets, F_F is distributed according to the F-distribution with 4 and 52 degrees of freedom: 35.812261, 3.92461, 95.888889, and 12.802532, respectively, which are all greater than the critical values of $F(4, 52)$ for $a = 0.05$ (The table of critical values can be

Table 3. Comparisons for NB-MCT versus NB, NB-MORT, NB-SMOTE, and NB-MC.

Dataset	NB-MCT	NB	NB-MORT	NB-SMOTE	NB-MC
adult	257.98 ± 43.06	$499.09 \pm 69.15\circ$	$382.72 \pm 56.88\circ$	$406.01 \pm 63.43\circ$	$371.70 \pm 0.46\circ$
bank	278.62 ± 38.18	$522.69 \pm 46.83\circ$	$435.92 \pm 50.78\circ$	$454.54 \pm 49.16\circ$	$398.90 \pm 0.30\circ$
breast-cancer	43.37 ± 19.87	$74.52 \pm 21.09\circ$	$58.39 \pm 22.02\circ$	$56.99 \pm 21.27\circ$	$20.10 \pm 0.30\bullet$
breast-w	7.95 ± 10.54	7.35 ± 10.04	6.00 ± 7.74	7.95 ± 10.41	$45.80 \pm 0.40\circ$
colic	48.28 ± 19.35	53.28 ± 20.99	51.70 ± 20.53	$56.06 \pm 21.43\circ$	$23.20 \pm 0.40\bullet$
colic.ORIG	31.17 ± 16.27	$80.33 \pm 26.67\circ$	$62.77 \pm 28.07\circ$	$73.94 \pm 25.34\circ$	24.40 ± 0.49
credit-g	82.60 ± 24.71	$232.81 \pm 38.72\circ$	$157.11 \pm 31.56\circ$	$189.67 \pm 36.50\circ$	70.00 ± 0.00
diabetes	79.12 ± 28.63	$152.80 \pm 39.91\circ$	$102.48 \pm 31.80\circ$	$110.26 \pm 35.86\circ$	$50.00 \pm 0.00\bullet$
hepatitis	9.18 ± 10.43	15.35 ± 12.70	14.44 ± 12.02	$16.45 \pm 12.66\circ$	12.30 ± 0.46
ionosphere	36.82 ± 20.40	31.63 ± 18.78	33.63 ± 19.70	37.06 ± 20.44	$22.50 \pm 0.50\bullet$
labor	2.68 ± 5.78	2.29 ± 5.78	2.20 ± 5.20	1.88 ± 4.88	3.70 ± 0.46
sick	74.77 ± 22.36	70.81 ± 28.53	66.14 ± 27.72	67.72 ± 27.36	$348.07 \pm 5.28\circ$
spect	22.41 ± 15.23	23.86 ± 16.04	24.37 ± 16.02	24.24 ± 16.13	21.20 ± 0.40
vote	19.49 ± 14.61	23.58 ± 16.26	21.62 ± 15.18	24.02 ± 16.18	26.70 ± 0.46
W/T/L	—	6/8/0	6/8/0	8/6/0	4/6/4

Table 4. Comparisons for C4.5-MCT versus C4.5, C4.5-MORT, C4.5-SMOTE, and C4.5-MC.

Dataset	C4.5-MCT	C4.5	C4.5-MORT	C4.5-SMOTE	C4.5-MC
adult	408.88 ± 67.79	$820.78 \pm 89.78 \circ$	$545.29 \pm 73.77 \circ$	$591.62 \pm 73.96 \circ$	$305.49 \pm 41.59 \bullet$
bank	312.87 ± 45.98	$581.54 \pm 62.22 \circ$	$439.66 \pm 58.30 \circ$	$446.19 \pm 55.85 \circ$	339.34 ± 57.24
breast-cancer	56.88 ± 18.60	$95.70 \pm 19.37 \circ$	$78.23 \pm 21.93 \circ$	$77.80 \pm 24.49 \circ$	$20.10 \pm 0.30 \bullet$
breast-w	21.91 ± 17.20	29.76 ± 22.26	24.22 ± 17.47	26.72 ± 20.54	17.03 ± 15.12
colic	49.21 ± 21.04	60.43 ± 24.37	55.01 ± 24.46	56.26 ± 23.90	$23.34 \pm 1.53 \bullet$
colic.ORIG	47.64 ± 19.62	$71.23 \pm 23.67 \circ$	62.51 ± 25.25	58.44 ± 22.46	$14.90 \pm 2.57 \bullet$
credit-g	174.36 ± 38.68	$271.86 \pm 43.02 \circ$	$214.17 \pm 42.41 \circ$	$215.70 \pm 40.45 \circ$	$71.09 \pm 4.29 \bullet$
diabetes	95.33 ± 31.78	$229.69 \pm 40.73 \circ$	$126.95 \pm 36.46 \circ$	$127.47 \pm 35.92 \circ$	$50.95 \pm 13.42 \bullet$
hepatitis	19.75 ± 12.52	28.55 ± 12.44	23.57 ± 13.37	26.15 ± 13.13	15.60 ± 7.71
ionosphere	35.07 ± 21.88	$54.66 \pm 22.11 \circ$	38.09 ± 21.58	37.51 ± 22.91	$20.92 \pm 12.54 \bullet$
labor	9.84 ± 9.41	10.52 ± 10.40	10.73 ± 9.87	12.68 ± 10.20	4.43 ± 3.79
sick	41.86 ± 23.63	$85.64 \pm 32.90 \circ$	$69.82 \pm 30.35 \circ$	$67.59 \pm 29.16 \circ$	48.09 ± 24.24
spect	27.30 ± 16.65	$45.05 \pm 19.44 \circ$	38.43 ± 19.31	$43.63 \pm 19.12 \circ$	22.75 ± 12.26
vote	9.19 ± 10.11	12.96 ± 12.12	12.00 ± 11.26	13.10 ± 12.42	9.05 ± 9.55
W/T/L	—	9/5/0	6/8/0	7/7/0	0/7/7

found in any statistical book.). So we reject the null hypotheses and proceed with Nemenyi and Bergmann tests to further analyze which pairs of algorithms are significantly different. Tables 7, 9, 11, and 13 respectively report the z-values and the p-values achieved on *post-hoc* tests for $\alpha = 0.05$, and also indicate which pairs of algorithms are significantly different. From the test results, we can see that our proposed MCT is also generally better than its competitors (MORT and SMOTE) and almost ties MC,[10] which is a state-of-the-art general method for making classifiers cost sensitive.

Table 14 shows the win/tie/loss counts of MCT versus its competitors under various cost ratios (5, 10, 15) in terms of the total misclassification costs. The experimental results show that MCT has better performance than its competitors

Table 5. Comparisons for SVM-MCT versus SVM, SVM-MORT, SVM-SMOTE, and SVM-MC.

Dataset	SVM-MCT	SVM	SVM-MORT	SVM-SMOTE	SVM-MC
adult	237.68 ± 36.07	$695.02 \pm 73.81 \circ$	$457.24 \pm 67.70 \circ$	$484.32 \pm 64.42 \circ$	$555.07 \pm 61.20 \circ$
bank	234.98 ± 44.79	$572.14 \pm 52.95 \circ$	$414.75 \pm 72.62 \circ$	$435.85 \pm 51.16 \circ$	$539.09 \pm 50.42 \circ$
breast-cancer	41.33 ± 16.72	$91.48 \pm 19.19 \circ$	$64.75 \pm 23.16 \circ$	$72.51 \pm 21.47 \circ$	54.08 ± 21.33
breast-w	27.35 ± 19.66	23.86 ± 17.81	25.46 ± 18.72	23.92 ± 17.17	$14.35 \pm 13.24 \bullet$
colic	50.78 ± 22.29	53.78 ± 24.61	54.23 ± 22.89	51.00 ± 25.28	38.11 ± 20.55
colic.ORIG	61.07 ± 26.51	$74.62 \pm 24.91 \circ$	68.88 ± 26.78	$74.77 \pm 23.64 \circ$	$46.90 \pm 22.02 \bullet$
credit-g	87.81 ± 23.56	$230.15 \pm 38.74 \circ$	$166.78 \pm 34.69 \circ$	$182.76 \pm 38.51 \circ$	$140.29 \pm 35.18 \circ$
diabetes	84.89 ± 32.08	$190.92 \pm 36.79 \circ$	$109.81 \pm 32.25 \circ$	$127.88 \pm 35.22 \circ$	104.29 ± 34.20
hepatitis	21.93 ± 12.70	24.25 ± 13.33	21.48 ± 12.43	23.88 ± 12.71	15.70 ± 11.74
ionosphere	35.37 ± 22.15	37.52 ± 22.41	37.52 ± 22.41	38.69 ± 21.57	28.02 ± 15.96
labor	5.33 ± 7.90	5.33 ± 7.90	5.33 ± 7.90	5.17 ± 7.86	4.76 ± 7.36
sick	36.43 ± 14.77	$62.59 \pm 31.21 \circ$	$52.77 \pm 26.29 \circ$	$53.39 \pm 26.38 \circ$	49.00 ± 25.88
spect	20.37 ± 12.79	$40.67 \pm 18.40 \circ$	$34.32 \pm 19.28 \circ$	$39.58 \pm 17.62 \circ$	26.84 ± 16.80
vote	9.61 ± 9.94	11.06 ± 11.78	9.04 ± 9.78	10.04 ± 10.55	9.38 ± 10.48
W/T/L	—	8/6/0	7/7/0	8/6/0	3/9/2

Table 6. Average rankings of KNN-MCT, KNN, KNN-MORT, KNN-SMOTE, and KNN-MC obtained by applying the Friedman test.

Algorithm	Ranking
KNN-MCT	1.5714
KNN	5
KNN-MORT	3.3571
KNN-SMOTE	3.1429
KNN-MC	1.9286

Table 7. KNN-MCT, KNN, KNN-MORT, KNN-SMOTE, and KNN-MC comparisons achieved on *post-hoc* tests for $\alpha = 0.05$.

i	Algorithms	$z = (R_0 - R_i)/SE$	p
10	KNN-MCT versus KNN	5.737097	0
9	KNN versus KNN-MC	5.139483	0
8	KNN versus KNN-SMOTE	3.107594	0.001886
7	KNN-MCT versus KNN-MORT	2.988072	0.002807
6	KNN versus KNN-MORT	2.749026	0.005977
5	KNN-MCT versus KNN-SMOTE	2.629503	0.008551
4	KNN-MORT versus KNN-MC	2.390457	0.016827
3	KNN-SMOTE versus KNN-MC	2.031889	0.042165
2	KNN-MCT versus KNN-MC	0.597614	0.550097
1	KNN-MORT versus KNN-SMOTE	0.358569	0.719918

Note: Nemenyi's procedure rejects those hypotheses that have an unadjusted p-value ≤ 0.005. Bergmann's procedure rejects these hypotheses:

- KNN-MCT versus KNN,
- KNN-MCT versus KNN-MORT,
- KNN-MCT versus KNN-SMOTE,
- KNN versus KNN-MORT,
- KNN versus KNN-SMOTE,
- KNN versus KNN-MC.

Table 8. Average rankings of NB-MCT, NB, NB-MORT, NB-SMOTE, and NB-MC obtained by applying the Friedman test.

Algorithm	Ranking
NB-MCT	2.25
NB	3.8571
NB-MORT	2.7857
NB-SMOTE	3.75
NB-MC	2.3571

Table 9. NB-MCT, NB, NB-MORT, NB-SMOTE, and NB-MC comparisons achieved on *post-hoc* tests for $\alpha = 0.05$.

i	Algorithms	$z = (R_0 - R_i)/SE$	p
10	NB-MCT versus NB	2.689264	0.007161
9	NB versys NB-MC	2.50998	0.012074
8	NB-MCT versus NB-SMOTE	2.50998	0.012074
7	NB-SMOTE versus NB-MC	2.330696	0.019769
6	NB versus NB-MORT	1.792843	0.072998
5	NB-MORT versus NB-SMOTE	1.613559	0.106623
4	NB-MCT versus NB-MORT	0.896421	0.370028
3	NB-MORT versus NB-MC	0.717137	0.473289
2	NB versus NB-SMOTE	0.179284	0.857714
1	NB-MCT versus NB-MC	0.179284	0.857714

Note: Nemenyi's procedure rejects those hypotheses that have an unadjusted p-value ≤ 0.005. Bergmann's procedure does not reject any hypotheses.

Table 10. Average rankings of C4.5-MCT, C4.5, C4.5-MORT, C4.5-SMOTE, and C4.5-MC obtained by applying the Friedman test.

Algorithm	Ranking
C4.5-MCT	1.8571
C4.5	4.7857
C4.5-MORT	3.3571
C4.5-SMOTE	3.8571
C4.5-MC	1.1429

Table 11. C4.5-MCT, C4.5, C4.5-MORT, C4.5-SMOTE, and C4.5-MC comparisons achieved on *post-hoc* tests for $\alpha = 0.05$.

i	Algorithms	$z = (R_0 - R_i)/SE$	p
10	C4.5 versus C4.5-MC	6.095666	0
9	C4.5-MCT versus C4.5	4.900437	0.000001
8	C4.5-SMOTE versus C4.5-MC	4.541869	0.000006
7	C4.5-MORT versus C4.5-MC	3.705209	0.000211
6	C4.5-MCT versus C4.5-SMOTE	3.34664	0.000818
5	C4.5-MCT versus C4.5-MORT	2.50998	0.012074
4	C4.5 versus C4.5-MORT	2.390457	0.016827
3	C4.5 versus C4.5-SMOTE	1.553797	0.120233
2	C4.5-MCT versus C4.5-MC	1.195229	0.231998
1	C4.5-MORT versus C4.5-SMOTE	0.83666	0.402784

Note: Nemenyi's procedure rejects those hypotheses that have an unadjusted p-value ≤ 0.005. Bergmann's procedure rejects these hypotheses:

- C4.5-MCT versus C4.5,
- C4.5-MCT versus C4.5-MORT,
- C4.5-MCT versus C4.5-SMOTE,
- C4.5 versus C4.5-MC,
- C4.5-MORT versus C4.5-MC,
- C4.5-SMOTE versus C4.5-MC.

Table 12. Average rankings of SVM-MCT, SVM, SVM-MORT, SVM-SMOTE, and SVM-MC obtained by applying the Friedman test.

Algorithm	Ranking
SVM-MCT	2
SVM	4.4643
SVM-MORT	2.9643
SVM-SMOTE	3.7143
SVM-MC	1.8571

Table 13. SVM-MCT, SVM, SVM-MORT, SVM-SMOTE, and SVM-MC comparisons achieved on *post-hoc* tests for $\alpha = 0.05$.

i	Algorithms	$z = (R_0 - R_i)/SE$	p
10	SVM versus SVM-MC	4.362584	0.000013
9	SVM-MCT versus SVM	4.123539	0.000037
8	SVM-SMOTE versus SVM-MC	3.107594	0.001886
7	SVM-MCT versus SVM-SMOTE	2.868549	0.004124
6	SVM versus SVM-MORT	2.50998	0.012074
5	SVM-MORT versus SVM-MC	1.852604	0.063939
4	SVM-MCT versus SVM-MORT	1.613559	0.106623
3	SVM versus SVM-SMOTE	1.25499	0.209482
2	SVM-MORT versus SVM-SMOTE	1.25499	0.209482
1	SVM-MCT versus SVM-MC	0.239046	0.81107

Note: Nemenyi's procedure rejects those hypotheses that have an unadjusted p-value ≤ 0.005. Bergmann's procedure rejects these hypotheses:

- SVM-MCT versus SVM,
- SVM-MCT versus SVM-SMOTE,
- SVM versus SVM-MORT,
- SVM versus SVM-MC,
- SVM-SMOTE versus SVM-MC.

Table 14. Comparisons of the total misclassification costs under various cost ratios.

Algorithms	Cost ratio = 5	Cost ratio = 10	Cost ratio = 15
KNN-MCT versus KNN	8/6/0	11/3/0	12/2/0
KNN-MCT versus KNN-MORT	6/6/2	8/6/0	10/4/0
KNN-MCT versus KNN-SMOTE	5/8/1	8/6/0	10/4/0
KNN-MCT versus KNN-MC	3/10/1	4/9/1	4/8/2
NB-MCT versus NB	6/7/1	6/7/1	6/8/0
NB-MCT versus NB-MORT	3/10/1	6/7/1	6/8/0
NB-MCT versus NB-SMOTE	4/9/1	7/6/1	8/6/0
NB-MCT versus NB-MC	3/10/1	2/10/2	4/6/4
C4.5-MCT versus C4.5	7/7/0	9/5/0	9/5/0
C4.5-MCT versus C4.5-MORT	3/11/0	6/8/0	6/8/0
C4.5-MCT versus C4.5-SMOTE	5/9/0	7/7/0	7/7/0
C4.5-MCT versus C4.5-MC	1/11/2	1/7/6	0/7/7
SVM-MCT versus SVM	5/9/0	6/8/0	8/6/0
SVM-MCT versus SVM-MORT	3/11/0	6/8/0	7/7/0
SVM-MCT versus SVM-SMOTE	4/10/0	6/8/0	8/6/0
SVM-MCT versus SVM-MC	3/11/0	3/9/2	3/9/2

(MORT and SMOTE) under various cost ratios. But the advantage is not so obvious when the cost ratios decreases. For example, when the cost ratio is 5, KNN-MCT slightly outperforms KNN-SMOTE only with five wins and one loss, NB-MCT slightly outperforms NB-MORT only with three wins and one loss, C4.5-MCT slightly outperforms C4.5-MORT only with three wins and zero loss, and SVM-MCT slightly outperforms SVM-MORT only with three wins and zero loss. Please also note that, our proposed MCT is competitive with the state-of-the-art MC[10] under various cost ratios except that the cost ratio is higher than 10, and yet at the same time C4.5 is chosen as the base classifier.

5. Discussion

Firstly, just as shown in Sec. 3, our current version simply clones each minority class instance according to the similarity between it and the single mode of the whole minority class instances. So, to some extend, we assume that the whole minority class instances comply with an unimodal distribution and thus the algorithm has a higher chance to work well when this assumption is satisfied. However, in the case that the data (or each feature of the data) has a bimodal or even more complex distribution, it is likely that only a (small) part of the data can be well represented by the constructed mode, or the mode might even belong to a different cluster. This may make the proposed MCT be sensitive to the data distribution. When the whole minority class instances do not comply with an unimodal distribution, the proposed algorithm might not be effective, and the predictive performance might be hindered. In order to diminish such sensitivity of MCT, we can construct multiple modes according to the centers of clusters, instead of always constructing a single mode for the whole minority class instances.

Secondly, for simplicity, our current version of MCT can only deal with categorical datasets, namely all attribute values are nominal. However, it is very easy to extend MCT for handling numeric attribute values or mixed attribute values just by replacing the mode of the minority class with the mean of the minority class. With regards to the measures of the similarity between each pair of instances, we can define them as inverse functions of the selected distance metrics such as Heterogeneous Euclidean-Overlap Metric (HEOM), Heterogeneous Value Difference Metric (HVDM),[38] Entropy-based Distance Metric,[6] and Neighborhood Counting Measure (NCM).[33]

At last, in order to make our compared results be replicable, the widely used total misclassification costs criterion and the commonly used misclassification cost matrix are adopted in our experiments. In the existing literatures,[12,24,35] the values of C_{FN} and C_{FP} are artificially assigned by users or domain experts. To our knowledge, we can learn an adaptive misclassification cost matrix according to the class distribution of training data. For example, we can simply define them as:

$$C_{\mathrm{FN}} = \frac{n_-}{n_+} \tag{3}$$

and

$$C_{\mathrm{FP}} = \frac{n_+}{n_-}, \tag{4}$$

where n_+ is the number of the positive instances and n_- is the number of the negative instances. Therefore, we can define the average misclassification costs criterion as:

$$\mathrm{Costs} = \frac{\mathrm{FN} \times C_{\mathrm{FN}} + \mathrm{FP} \times C_{\mathrm{FP}}}{n_+ + n_-}. \tag{5}$$

Obviously, when n_+ is equal to n_-, our defined average misclassification costs criterion is just the error rate criterion (the rate of test instances incorrectly classified) and thus the class-imbalanced cost-sensitive learning is transferred into the traditional class-balanced cost-insensitive learning.

6. Conclusion

In order to deal with the class-imbalanced cost-sensitive classification problems, we propose a novel MCT, which alters the class distribution of training data by cloning each minority class instance according to the similarity between it and the mode of the minority class. The experimental results based on paired t-tests and nonparametric Friedman tests show that our proposed MCT performs much better than MORT and SMOTE in terms of the total misclassification costs.

Besides, we discuss some related issues: (1) how to diminish the sensitivity of MCT; (2) how to extend MCT for handling numeric attribute values or mixed attribute values; (3) how to learn an adaptive misclassification cost matrix according to the class distribution of training data.

Acknowledgments

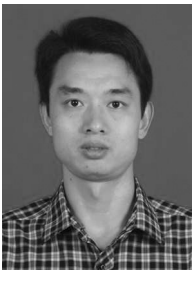
The work was partially supported by the National Natural Science Foundation of China (61203287), the Program for New Century Excellent Talents in University (NCET-12-0953), the Chenguang Program of Science and Technology of Wuhan (2015070404010202), and the Fundamental Research Funds for the Central Universities (CUG130504, CUG130414).

References

1. S. Barua, M. Islam, X. Yao and K. Murase, MWMOTE-Majority weighted minority oversampling technique for imbalanced data set learning, *IEEE Trans. Knowl. Data Eng.* **26**(2) (2014) 405–425.
2. P. K. Chan and S. J. Stolfo, Toward scalable learning with non-uniform class and cost distributions: A case study in credit card fraud detection, in *Proc. Fourth Int. Conf. Knowledge Discovery and Data Mining* (1998) 164–168.
3. N. V. Chawla, Data mining for imbalanced datasets: An overview, in *Data Mining and Knowledge Discovery Handbook* Vol. 5 (Springer, US, 2006), pp. 853–867.

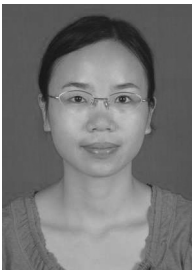
4. N. V. Chawla, K. W. Bowyer, L. O. Hall and W. P. Kegelmeyer, SMOTE: Synthetic minority over-sampling Technique, *J. Artif. Intell. Res.* **16** (2002) 321–357.
5. N. V. Chawla, A. Lazarevic, L. O. Hall and K. W. Bowyer, SMOTEBoost: Improving prediction of the minority class in boosting, in *Proc. Seventh European Conf. Principles and Practice of Knowledge Discovery in Databases* (2003) pp. 107–119.
6. J. G. Cleary and L. E. Trigg, K*: An instance-based learner using an entropic distance measure, in *Proc. 12th Int. Machine Learning Conf.*, Tahoe City, CA (Morgan Kaufmann, 1995), pp. 108–114.
7. S. Cost and S. Salzberg, A weighted nearest neighbor algorithm for learning with symbolic features, *Mach. Learn.* **10**(1) (1993) 57–78.
8. M. Dash and H. Liu, Feature selection for classification, *Intell. Data Anal.* **1**(3) (1997) 131–156.
9. J. Demasal, Statistical comparisons of classifiers over multiple data sets, *J. Mach. Learn. Res.* **7** (2006) 1–30.
10. P. Domingos, Metacost: A general method for making classifiers cost432 sensitive, in *Proc. Fifth ACM SIGKDD Int. Conf. Knowledge discovery and data mining* (ACM, 1999), pp. 155–164.
11. T. Fawcett and F. Provost, Adaptive fraud detection, *Data Min. Knowl. Discov.* **1**(3) (1997) 291–316.
12. Y. Guo, H. Zhang and B. Spencer, Cost-Sensitive Self-Training, in *Proc. 25th Canadian Conf. Artificial Intelligence*, LNAI, Vol. 7310 (2012), pp. 74–84.
13. H. He and E. Garcia, Learning from imbalanced data, *IEEE Trans. Knowl. Data Eng.* **21**(9) (2009) 1263–1284.
14. N. Japkowicz and S. Stephen, The class imbalance problem: A systematic study, *Intell. Data Anal.* **6**(5) (2002) 429–450.
15. L. Jiang, Z. Cai and D. Wang, Improving Naive Bayes for classification, *Int. J. Comput. Appl.* **32**(3) (2010) 328–332.
16. L. Jiang, Z. Cai, H. Zhang and D. Wang, Not so greedy: Randomly selected Naive Bayes, *Expert Syst. Appl.* **39**(12) (2012) 11022–11028.
17. L. Jiang, C. Li, Z. Cai and H. Zhang, Sampled Bayesian network classifiers for class-imbalance and cost-sensitive learning, in *Proc. 25th IEEE Int. Conf. Tools with Artificial Intelligence (ICTAI'13)*, Herndon, Virginia, USA (2013), pp. 512–517.
18. L. Jiang, D. Wang, H. Zhang, Z. Cai and B. Huang, Using instance cloning to improve Naive Bayes for ranking, *Int. J. Pattern Recogn. Artif. Intell.* **22**(6) (2008) 1121–1140.
19. L. Jiang, H. Zhang and Z. Cai, Discriminatively improving Naive Bayes by evolutionary feature selection, *Rom. J. Inform. Sci. Technol.* **9**(3) (2006) 163–174.
20. S. S. Keerthi, S. K. Shevade, C. Bhattacharyya and K. R. K. Murthy, Improvements to Platt's SMO algorithm for SVM classifier design, Technical Report CD-99-14, Department of Mechanical and Production Engineering, National University of Singapore, Singapore (1999).
21. P. C. R. Lane, D. Clarke and P. Hender, On developing robust models for favourability analysis: Model choice, feature sets and imbalanced data, *Decis. Support Syst.* **53** (2012) 712–718.
22. P. Langley, W. Iba and K. Thomas, An analysis of Bayesian classifiers, in *Proc. Tenth National Conf. Artificial Intelligence* (AAAI Press, 1992), pp. 223–228.
23. D. Lewis and W. A. Gale, A sequential algorithm for training text classifiers, in *Association for Computing Machinery-Special Interest Group on Information Retrieval* (1994), pp. 3–12.
24. Y. Li, J. Kwok and Z. Zhou, Cost-sensitive semi-supervised support vector machine, in *Proc. 24th AAAI Conf. Artificial Intelligence* (2010), pp. 500–505.

25. C. X. Ling and V. S. Sheng, Cost-sensitive learning and the class imbalance problem, in *Encyclopedia of Machine Learning* (Springer, US, 2008), pp. 231–235.
26. X. Y. Liu, J. Wu and Z. H. Zhou, Exploratory undersampling for class-imbalance learning, *IEEE Trans. Syst. Man, Cybern. B, Cybern.* **39**(2) (2009) 539–550.
27. C. Merz, P. Murphy and D. Aha, UCI repository of machine learning databases, Department of ICS, University of California, Irvine (1997), Available at http://www.ics.uci.edu/mlearn/MLRepository.html.
28. C. Nadeau and Y. Bengio, Inference for the generalization error, *Mach. Learn.* **52**(3) (2003) 239–281.
29. J. Platt, Fast training of support vector machines using sequential minimal optimization, in *Advances in Kernel Methods. Support Vector Learning*, eds. B. Scholkopf, C. J. C. Burges and A. J. Smola (MIT Press, Cambridge, MA, 1999), pp. 185–208.
30. J. R. Quinlan, *C4.5: Programs for Machine Learning* (Morgan Kaufmann, San Mateo, CA, 1993).
31. P. N. Tan, M. Steinbach and V. Kumar, *Introduction to Data Mining* (Pearson Addison-Wesley, 2006).
32. V. Vapnik, *The Nature of Statistical Learning Theory* (Springer, New York, 1995).
33. H. Wang, Nearest neighbors by neighborhood counting, *IEEE Trans. Pattern Anal. Mach. Intell.* **28**(6) (2006) 942–953.
34. J. Wang, P. Neskovic and L. N. Cooper, Improving nearest neighbor rule with a simple adaptive distance measure, *Pattern Recogn. Lett.* **28**(2) (2007) 207–213.
35. T. Wang, Z. Qin, S. Zhang and C. Zhang, Cost-sensitive classification with inadequate labeled data, *Inform. Syst.* **37** (2012) 508–516.
36. G. Weiss and F. Provost, Learning when training data are costly: The effect of class distribution on tree induction, *J. Artif. Intell. Res.* **19** (2003) 315–354.
37. D. L. Wilson, Asymptotic properties of nearest neighbor rules using edited data, *IEEE Trans. Syst., Man Cybern.* **2**(3) (1972) 408–421.
38. D. R. Wilson and T. R. Martinez, Improved heterogeneous distance functions, *J. Artif. Intell. Res.* **6**(1) (1997) 1–34.
39. I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd edn. (Morgan Kaufmann, San Francisco, 2005).
40. X. Wu, V. Kumar, J. R. Quinlan *et al.*, Top 10 algorithms in data mining, *Knowl. Inform. Syst.* **14**(1) (2008) 1–37.



Liangxiao Jiang received his Ph.D. from the China University of Geosciences in June 2009. Currently, he is a Professor at the Department of Computer Science, China University of Geosciences. His research interests include data mining and machine learning.

Chen Qiu received her B.Sc. degree in June 2014. Currently, she is an M.Sc. student in the Department of Computer Science, China University of Geosciences. Her research interests include data mining and machine learning.



Chaoqun Li received her Ph.D. from the China University of Geosciences in June 2012. Currently, she is a Lecturer at the Department of Mathematics, China University of Geosciences. Her research interests include data mining and machine learning.