2014 International Joint Conference on Neural Networks (IJCNN)
July 6-11, 2014, Beijing, China

Learning from Combination of Data Chunks for Multi-class Imbalanced Data

Xu-Ying Liu[1,2] and Qian-Qian Li[1]
[1] Key Laboratory of Computer Network and Information Integration, MOE
School of Computer Science and Engineering, Southeast University, Nanjing, China
[2] National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China
Email: {liuxy, liqianqian}@seu.edu.cn

***Abstract*—Class-imbalance is very common in real-world applications. Previous studies focused on binary-class imbalance problem, whereas multi-class imbalance problem is more general and more challenging. Under-sampling is an effective and efficient method for binary-class imbalanced data. But when it is used for multi-class imbalanced data, many more majority class examples are ignored because there are often multiple majority classes, and the minority class often has few data. To utilize the information contained in the majority class examples ignored by under-sampling, this paper proposes a method ChunkCombine. For each majority class, it performs under-sampling multiple times to obtained non-overlapping data chunks, such that they contain the most information that a data sample of the same size can contain. Each data chunk has the same size as the minority class to achieve balance. Then every possible combination of the minority class and each data chunk from every majority class forms a balanced training set. ChunkCombine uses ensemble techniques to learn from the different training sets derived from all the possible combinations. Experimental results show it is better than many other popular methods for multi-class imbalanced data when average accuracy, G-mean and MAUC are used as evaluation measures. Besides, we discuss different evaluation measures and suggest that, a multi-class F-measure Mean F-Measure (MFM) is unsuitable for multi-class imbalanced data in many situations because it is not consistent with the standard F-measure in binary-class case and it is close to accuracy.**

I. Introduction

Class-imbalance problem exists in many real-world applications, that is, some classes have much smaller populations than the other classes, and the examples of the smaller classes are more important. The smaller classes are called the "minority classes" and the larger classes are called the "majority classes". For example, in face detection problem, an image normally contains about 10^5 windows with different locations and scales, and the non-face windows are about 10^4 times more than the face-windows, where the latter are targets. Standard machine learning methods assume that all examples have the same importance and tend to overlook the minority class examples to achieve high accuracy. In the above example, the accuracy of the classifier always predicting non-face window is 99.99%, but obviously it is useless since it cannot detect any face. Therefore, accuracy is not an appropriate evaluation measure for class-imbalance problem, and instead, F-measure, G-mean and AUC [1] are used for evaluation.

Class-imbalance learning has been considered as one of the most challenging problems in machine learning and data mining [2], whereas previous studies mainly focused on binary-class data [3], [4], [5], [6], [7], [8], [9]. It is noteworthy that multi-class imbalance is more general and more challenging than binary-class imbalance. The difficulty of multi-class imbalance problem includes but not limited to the following aspects: multi-class imbalance is multilateral, which is more complicated than the bilateral imbalance in binary-class case; the degree of concept complexity is much higher; there are more overlapping areas among classes; the imbalance rates (the size of a majority class divided by the size of the smallest class) maybe very high; and the smallest class may has very few data. However, the studies on multi-class imbalance are relatively limited, which will be briefly reviewed in Section II.

Sampling, including over-sampling and under-sampling, is one of the most popular and effective methods for binary-class imbalance. Random over-sampling samples from the minority class to make data balanced. It takes the risk of over-fitting since it replicates the minority class examples. For multi-class imbalance, all the classes except the largest class are over-sampled to achieve balance, which greatly increases the chance of over-fitting. Efficiency is important to multi-class classification, while over-sampling is inefficient. Under-sampling samples from the majority classes to make data balanced. Obviously, it is much more efficient. But it ignores too many majority class examples, especially when there are multiple majority classes, and when the smallest class has very few data, so the performance would be very poor, and quite unstable for different training sets since the sampled data is a small fraction of the original data. *Therefore, it is important to utilize the information contained in the majority class examples ignored by under-sampling to effectively handle multi-class imbalanced data.*

Since the performance of under-sampling would be very poor, it is nature to assume that the trained classifier is a weak learner, that is, its performance is just slightly better than random guess [19], [20]. Ensemble techniques [20] have shown their successes in improving the generalization ability of weak learners. In this paper, we use ensemble techniques to utilize the information contained in the ignored majority class examples. Ensemble methods not only can reduce the bias which measures how closely the average estimate of the learning algorithm is able to approximate the target, but also can reduce the variance which measures how much the estimate of the learning algorithm fluctuates for different training sets of the same size [21]. It is well known that Boosting, such as AdaBoost [22], primarily reduces the bias [23], [24], and Bagging has tremendous variance reduction effect [25]. Thus, both of Boosting and Bagging will help

978-1-4799-1484-5/14/$31.00 ©2014 IEEE

improving under-sampling to handle multi-class imbalanced data.

Our proposed ChunkCombine method utilizes both of Boosting and Bagging-like ensemble techniques to utilize the information contained in the majority class examples ignored by under-sampling. For each majority class, it performs under-sampling multiple times to obtained non-overlapping data chunks, such that they contain the most information that a data sample of the same size can contain. Each data chunk has the same size as the minority class to achieve balance. Then every possible combination of the minority class and each data chunk from every majority class forms a balanced training set. ChunkCombine learns a set of multi-class AdaBoost classifiers from the different training sets derived from all the possible combinations. Finally, all the weak learners are combined by hard voting. Experimental results show ChunkCombine is better than many other popular methods for multi-class imbalanced data when average accuracy, G-mean and MAUC are used as evaluation measures.

Besides, we discuss some popular evaluation measures for multi-class imbalance learning, including G-mean, average accuracy AvgAcc [26], MAUC [27] and Mean F-Measure (MFM) [28], and conclude that MFM is not identical to F-measure in binary-class case, while all the other measures are consistent with their binary-class versions. Methods having higher G-mean and AvgAcc values can have much lower MFM values. It should be cautious to use MFM as evaluation measure for multi-class imbalance learning, especially when there are "multi-majority" classes [14].

The rest of the paper is organized as follows. Section II briefly introduces the related work. Section III describes the proposed ChunkCombine method. Section IV discusses the evaluation measures. Section V shows the experiments and Section VI concludes.

II. Related Work

A. Methods for Binary-class Imbalance

Methods for binary-class imbalance can be roughly grouped into 5 categories. (1) Balancing the training set via sampling. Sampling methods include over-sampling and under-sampling. Besides random over-sampling, SMOTE [3] is a very popular over-sampling method. To avoid over-fitting of over-sampling, SMOTE adds synthetic examples to the minority classes by interpolating between the minority class examples and their neighbors belonging to the same class. SMOTE takes the risk of introducing noise, which can be greatly enhanced when there are multiple classes. (2) Cost-sensitive learning methods [29], [30]. They assign higher and lower costs to the minority and majority class examples respectively to convert a binary-class imbalance problem to a cost-sensitive problem, and then train a classifier to minimize total cost. (3) Refining the placement of decision boundary. The most straightforward way is threshold-moving, which moves the decision threshold such that the minority class examples are easier to be classified correctly. (4) Recognition-based methods, such as one-class SVM [31]. The model is created based on the target class alone. To classify an instance, the similarity of the instance and the target class is measured, and the instance will be predicted as a target class example when the similarity exceeds threshold. (5) Methods particularly designed for class-imbalance problem.

There are many ensemble based methods for binary-class imbalanced data. To improve under-sampling, Chan and Stolfo [32] splits the majority classes into several data chunks with size similar to the minority class, and trains SVMs from the combinations of the minority class and each of the data chunk of the majority class, then combines the SVMs using stacking to minimize error rate; EasyEnsemble [7] explores the information contained in the majority class examples ignored by under-sampling by running under-sampling procedure multiple times independently, in each procedure an AdaBoost classifier is trained, finally, all the weak learners of all the AdaBoost classifiers are combined by soft ensemble; BalanceCascade [7] explores the information contained in the majority class in a cascade-style, after each under-sampling procedure, the majority class examples correctly classified by the Adaboost classifier trained in the latest under-sampling procedure are considered as redundant and are removed from the majority class, then like EasyEnsemble, all the weak learners of all the AdaBoost classifiers are combined by soft ensemble.

SMOTEBoost [33] is proposed to improve SMOTE, in each boosting round, the weight distribution is adjusted by SMOTE to focus more on the minority class examples, thus, the weak learners that could be affected by noise introduced by SMOTE can be boosted into strong learners. It can not only be used in binary-class classification, but also in multi-class classification. There are many other ensemble methods for binary-class imbalanced data, such as RUSBoost [34], and cost-sensitive ensemble methods, such as AsymBoost [35]. Please refer to [36] for detailed literature review.

B. Methods for Multi-class Imbalance

There are several work on multi-class imbalanced data [10], [11], [12], [13], [14], [15], [16], [37]. [11] uses multi-class cost-sensitive method to handle multi-class imbalanced data, where the optimal cost vector is determined by a genetic algorithm. MC-HDDT [13] is a decision tree method which uses a multi-class splitting criterion in favor of the minority classes. DyS [16] dynamically samples data for neural network to update the weights during the learning procedure. [14] categories the multi-class imbalance problems to the problems with "multi-minority" and "multi-majority" classes, and proposes OvNC method which combines boosting and over-sampling. To find the best factor of cost-sensitive learning and sampling methods for multi-class imbalanced data, [18] proposes an effective wrapper framework incorporating the evaluation measure into the objective function of cost sensitive learning as well as re-sampling directly. EasyEnsemble.M [37] extends the EasyEnsemble method to multi-class case. Like EasyEnsemble, it performs under-sampling multiple times independently to obtain several balanced training sets, and trains a multi-class AdaBoost classifier using AdaBoost.M [38] algorithm from each of them. Finally, all the weak learners of the AdaBoost classifiers are combined via hard voting.

There are some decomposition-based methods, including OVA (one versus all) and OVO (one versus one) based methods, such as [10] uses OVA and OVO reductions and designs a decision rule to handle the binary-class imbalance in each

subtask. OAHO [12] reduces a multi-class learning task to a series of binary-class subtasks by grouping the minority classes together to decrease the imbalance rate. Recently, Liu et al. proposed an error correcting output codes method imECOC [17], in each dichotomy, both of between-class and within-class imbalance are handled; and each dichotomy is assigned a weight to reflect its importance in decoding stage, a test instance will be classified to the class with the minimal weighted distance; the optimal weights are obtained by minimizing a weighted loss function favoring the minority classes by using large margin method.

III. ChunkCombine Method

The basic idea of our proposed ChunkCombine method is that, to utilize the information contained in the majority class examples ignored by under-sampling, for each majority class, ChunkCombine performs under-sampling without replacement multiple times to obtained non-overlapping data chunks (data samples), such that they contain the most information that a data sample of the same size can contain. Each data chunk has the same size as the minority class to achieve balance. Then for every possible combination of the minority class and each data chunk from every majority class, which forms a balanced training set, ChunkCombine trains a multi-class AdaBoost classifier using AdaBoost.M [38]. Finally, a set of AdaBoost classifiers are obtained, and all the weak learners in them are combined via hard voting.

Suppose there are k classes, $D = \{D_1, D_2, \ldots, D_k\}$ with increasing class sizes $n_1 \leq n_2 \leq \cdots \leq n_k$, D_1 is the minority class, and the other classes are called the majority classes. For a majority class $D_i, i \neq 1$, ChunkCombine samples a set of non-overlapping data chunks $D_iC = \{D_iC_1, \ldots, D_iC_{m_i}\}$ by performing under-sampling without replacement multiple times, where m_i is the number of data chunks in class D_i. Thus, all the examples in D_iC are unique, such that the data chunks contain the most information that a data sample of the same size can contain. To achieve balance, all the data chunks have the same size as the minority class, that is, $|Q| = n_1, \forall Q \in D_iC$. It is obvious that, $n_1 m_i \leq n_i$ should hold since all the examples in the data chunks are unique. To obtain the non-overlapping data chunks, the under-sampling procedures are dependent, after a data chunk is sampled from D_i, it is removed from the class, and the following data chunk is sampled from the remaining examples. ChunkCombine repeats the above process for every majority class $D_i, i = 2, \ldots, k$.

The combination of the minority class and a data chunk from every majority class forms a balanced training set $\{D_1, Q_i, \ldots, Q_k\}$, with $Q_i \in D_iC, i = 2, \ldots, k$, then ChunkCombine trains a multi-class AdaBoost classifier from it using AdaBoost.M. There are altogether $\Pi_{i=2}^{k} m_i$ different training sets derived from all possible combinations, that is, $\{\{D_1, D_2C_{i_2}, \ldots, D_kC_{i_k}\} | i_j = 1, \ldots, m_j, j = 2, \ldots, k\}$, so we get $\Pi_{i=2}^{k} m_i$ AdaBoost classifiers. Suppose the multi-class learning algorithm $L : X \to Y$ is used to train the weak learners of AdaBoost, where, X is the input space and Y is the output space, $Y \in \{1, \ldots, k\}$. The AdaBoost classifier trained from the training set $\{D_1, D_2C_{i_2}, \ldots, D_kC_{i_k}\}$ is denoted by $A_{i_2,\ldots,i_k}$. It has T weak learners $\{h^{(t)}_{i_2,\ldots,i_k}\}_{t=1}^{T}$ and the corresponding weights $\{\alpha^{(t)}_{i_2,\ldots,i_k}\}_{t=1}^{T}$. Finally, all the weak learners of the AdaBoost classifiers $\{h^{(t)}_{i_2,\ldots,i_k} | i_j = 1, \ldots, m_j, j = 2, \ldots, k, t = 1, \ldots, T\}$ are combined to form an ensemble via hard voting, that is,

$$H(x) = \arg_y \max \sum_{i_2,\ldots,i_k} \sum_{t=1}^{T} I(h^{(t)}_{i_2,\ldots,i_k}(x) = y), \quad (1)$$

where, $I(x) = 1$ if x is true and 0 otherwise. The output of ChunkCombine is a single ensemble though it looks like an "ensemble of ensemble". It combines all the weak learners in all the AdaBoost classifiers instead of the AdaBoost classifiers. An alternative view of the weak learners is to treat them as features that are extracted by the ensemble learning method and can only take discrete values [39]. In this viewpoint, AdaBoost classifier is simply a linear classifier built on these features. Features extracted from different training sets thus contain information of different aspects of the original data set. Finally, instead of counting votes from the AdaBoost classifiers $\{A_{i_2,\ldots,i_k} | i_j = 1, \ldots, m_j, j = 2, \ldots, k\}$, we collect all the features $\{h^{(t)}_{i_2,\ldots,i_k} | i_j = 1, \ldots, m_j, j = 2, \ldots, k, t = 1, \ldots, T\}$ and form an ensemble classifier from them. Hard voting instead of soft voting is used to form the ensemble because when under-sampling ignores too many majority class examples, even the AdaBoost classifier could be unstable, thus its base learners' weights which are computed by the function of error rates are also unstable, so it is better not to use the unreliable weights for soft voting, and instead to use discrete values for hard voting. The pseudo code is shown in algorithm 1. It is worth noting that:

(1) Not all the majority class examples are used in ChunkCombine, otherwise, it will be extremely time consuming and it is unnecessary when the union of all data chunks are representative enough. In binary-class case, [7] has shown that it is unnecessary to use all the majority class examples in the similar methods (ensemble methods trying to utilize the information ignored by under-sampling), such as EasyEnsemble [7].

(2) The data chunks of each majority class are non-overlapping, so all the examples in them are unique, such that the data chunks contains the most information that a data sample with the same size can contain. It is different from EasyEnsemble.M [37], in the latter the under-sampling procedures are independent, so the samples of a class in different training sets can have overlaps, which will reduce the amount of information contained in different training sets.

(3) And it is not like BalanceCascade in binary-class case. BalanceCascade removes the majority class examples correctly classified by the AdaBoost classifier after each under-sampling procedure. So the data distribution changes considerably after the examples are removed. This maybe the reason that it is not as good as EasyEnsemble. Though ChunkCombine also removes some majority class examples after each under-sampling procedure, the removed data are not biased, they are randomly sampled data. When there remains much more majority class examples than the minority class, the change of the data distribution is mere.

(4) In general, under-sampling ignores much more majority class examples in multi-class imbalanced data than the binary-class imbalanced data, especially when there are just few

Algorithm 1 The ChunkCombine algorithm.

{Input: Data set $D = \{D_1, D_2, \ldots, D_k\}$ with increasing class sizes $n_1 \leq n_2 \leq \cdots \leq n_k$, k the number of classes, D_i the i-th class, X the input space, Y the output space, $Y \in \{1, \ldots, k\}$, m_i the number of data chunk in the majority class $D_i, i = 2, \ldots, k$, and $n_1 m_i \leq n_i, i = 2, \ldots, n$ should hold, T the number of iterations of AdaBoost, multi-class learning method $L : X \rightarrow Y$ to train weak learners}
for $i = 1$ to k **do**
  $D_iC \leftarrow \emptyset$ {the set of data chunks of D_i}
  for $j = 1$ to m_i **do**
    Sample data chunk D_iC_j, with $|D_iC_j| = n_1$, from D_i using under-sampling without replacement
    $D_iC \leftarrow D_iC \cup D_iC_j$
    $D_i \leftarrow D_i - D_iC_j$ {remove the sampled data from the class}
  end for
end for
for $i_2 = 2$ to m_2 **do**
  .....
  for $i_k = 1$ to m_k **do**
    Learn a multi-class AdaBoost classifier $A_{i_2,\ldots,i_k}$ using AdaBoost.M algorithm from the balanced training set $\{D_1, D_2C_{i_2}, \ldots, D_kC_{i_k}\}$, which has T weak learners $\{h^{(t)}_{i_2,\ldots,i_k}\}^T_{t=1}$ and corresponding weights $\{\alpha^{(t)}_{i_2,\ldots,i_k}\}^T_{t=1}$, i.e, $A_{i_2,\ldots,i_k}(x) = \arg_y \max \sum_{t=1}^T \alpha^{(t)}_{i_2,\ldots,i_k} I(h^{(t)}_{i_2,\ldots,i_k}(x) = y)$, where, $I(x) = 1$ if x is true and 0 otherwise.
  end for
  .....
end for
Output: An ensemble $H(x) = \arg_y \max \sum_{i_2,\ldots,i_k} \sum_{t=1}^T I(h^{(t)}_{i_2,\ldots,i_k}(x) = y)$.

examples in the minority class. So the performance could be very poor and quite unstable for different training sets. ChunkCombine uses AdaBoost to train classifiers because it can effectively reduce the bias. All possible combinations of the minority class and each data chunk of every majority class are performed to form different training sets for learning in order to further reduce the bias and the variance of ensemble. It is quite different from EasyEnsemble.M [37]. Section V shows its effectiveness by comparing it with EasyEnsemble.M.

(5) ChunkCombine uses multiple ensemble techniques. Some previous studies combine different ensemble techniques and achieve stronger generalization, such as MultiBoost [40] combines boosting with bagging, Cocktail Ensemble [41], and EasyEnsemble.

IV. Discussion on Evaluation Measures

A. Binary-class Evaluation Measures

It is well known that accuracy is not appropriate for evaluation in imbalance learning no matter there are two or more classes. F-measure, G-mean and AUC (Area Under the ROC Curve) [1] are popular evaluation measures for binary-class imbalance learning methods. F-measure and G-mean are functions of confusion matrix as shown in Table I, which are defined as follows. Here, the minority class is the positive class (or class 1) since it is more important and the majority class is the negative class (or class 2).

$$\begin{aligned} Acc_+ &= \frac{TP}{TP+FN} = \frac{TP}{n_+}, \\ Acc_- &= \frac{TN}{TN+FP} = \frac{TN}{n_-}, \\ \text{G-mean} &= (Acc_+ \times Acc_-)^{\frac{1}{2}}, \quad (2) \\ \text{AvgAcc} &= \frac{1}{2}(Acc_+ + Acc_-), \quad (3) \end{aligned}$$

where, n_+, n_- is the size of the positive and negative class, respectively, G-mean is the geometric mean of the accuracy of each class. We also list the arithmetic mean i.e., AvgAcc here for extension to multi-class case later.

$$\begin{aligned} P &= \frac{TP}{TP+FP}, \\ R &= \frac{TP}{TP+FN} = Acc_+, \\ \text{F1} &= \frac{2}{\frac{1}{P}+\frac{1}{R}} = \frac{2PR}{P+R}, \quad (4) \end{aligned}$$

where, P is precision which measures how many examples in the predicted positives are correctly classified (as true positives), and R is recall which measures how many examples in the actual positives are correctly classified (as true positives). F1 is the harmonic mean of precision and recall. And AUC is defined as:

$$\text{AUC} = \frac{1}{n_+ n_-} \sum_{x_+ \in D_+, x_- \in D_-} I(f(x_+) \geq f(x_-)), \quad (5)$$

where, $I(x) = 1$ is x is true and 0 otherwise, D_+, D_- is the positive and negative class, respectively, and $f(x)$ outputs real values for predicting the positive class. AUC measures the probability of a randomly drawn positive instance has higher f value than a randomly drawn negative instance.

B. Multi-class Evaluation Measures

For multi-class imbalance, suppose there are k classes and they are ordered by increasing class size $n_1 \leq \cdots \leq n_k$, and $cm_{k\times k}$ is the confusion matrix with $cm(i,j)$ indicates the number of the examples of class i predicted to class j. In the studies on multi-class imbalanced data, several evaluation measures are used, including the multi-class G-mean used in [11], [14], [16], [17], [37], which is the generalization of G-mean in Eq. (2), and average accuracy AvgAcc [26] used in [15], which is the multi-class generalization of AvgAcc in Eq. (3). They are defined as:

$$\begin{aligned} Acc_i &= \frac{cm(i,i)}{n_i}, \\ \text{G-mean} &= (\Pi_{i=1}^k Acc_i)^{\frac{1}{k}}, \quad (6) \\ \text{AvgAcc} &= \frac{1}{k}\sum_{i=1}^k Acc_i, \quad (7) \end{aligned}$$

TABLE I. CONFUSION MATRIX OF BINARY-CLASS CASE (THE MINORITY CLASS IS THE POSITIVE CLASS).

	Predicted Positive Class	Predicted Negative Class
Actual Positive Class	TP (True Positives)	FN (False Negatives)
Actual Negative Class	FP (False Positives)	TN (True Negatives)

TABLE II. EXAMPLE OF COMPARING DIFFERENT EVALUATION MEASURES (BOLD FACE INDICATES THE PERFORMANCE IS BETTER THAN OR EQUAL TO THE PERFORMANCE OF cm_0).

	cm_0 $\begin{bmatrix} 3 & 7 \\ 10 & 90 \end{bmatrix}$	cm_1 $\begin{bmatrix} 6 & 4 \\ 80 & 20 \end{bmatrix}$	cm_2 $\begin{bmatrix} 6 & 4 \\ 55 & 45 \end{bmatrix}$	cm_3 $\begin{bmatrix} 6 & 4 \\ 40 & 60 \end{bmatrix}$	cm_4 $\begin{bmatrix} 6 & 4 \\ 30 & 70 \end{bmatrix}$	cm_5 $\begin{bmatrix} 6 & 4 \\ 20 & 80 \end{bmatrix}$	cm_6 $\begin{bmatrix} 6 & 4 \\ 13 & 87 \end{bmatrix}$	cm_7 $\begin{bmatrix} 6 & 4 \\ 10 & 90 \end{bmatrix}$
Acc1	0.3	0.6	0.6	0.6	0.6	0.6	0.6	0.6
Acc2	0.9	0.2	0.45	0.6	0.7	0.8	0.87	0.9
G-mean	0.52	0.35	**0.52**	**0.60**	**0.65**	**0.69**	**0.72**	**0.73**
AvgAcc	0.60	0.40	0.53	**0.60**	**0.65**	**0.70**	**0.74**	**0.75**
F_1(F)	0.26	0.13	0.17	0.21	**0.26**	**0.33**	**0.41**	**0.46**
F_2	0.91	0.32	0.60	0.73	0.80	0.87	**0.91**	**0.93**
MFM	0.59	0.22	0.39	0.47	0.53	**0.60**	**0.66**	**0.69**
Acc	0.85	0.24	0.46	0.60	0.69	0.78	**0.85**	**0.87**

where, Acc_i is the accuracy of class i. Mean F-measure (MFM) [28] used in [11], [17], which is defined as:

$$\begin{aligned} R_i &= Acc_i = \frac{cm(i,i)}{n_i}, \\ P_i &= \frac{cm(i,i)}{\sum_{j=1}^{k} cm(j,i)}, \\ F_i &= \frac{2P_iR_i}{P_i+R_i}, \\ \text{MFM} &= \frac{1}{k}\sum_{i=1}^{k} F_i, \end{aligned} \tag{8}$$

and multi-class AUC (MAUC) [27] used in [14], [16], [17], which is defined as:

$$\begin{aligned} A(i,j) &= (A(i|j) + A(j|i))/2, \\ \text{MAUC} &= \frac{2}{k(k-1)}\sum_{i<j} A(i,j), \end{aligned} \tag{9}$$

where, $A(i|j)$ is the probability that a randomly drawn member of class i will have a lower estimated probability of belonging to class j than a randomly drawn member of class j. It should be noted that $A(i|j) \neq A(j|i)$.

It is easy to see that, when $k = 2$, multi-class G-mean and AvgAcc are identical to their definitions in the binary-class case. And when $k = 2$, $A(i|j) = A(j|i)$ since $p(y = 2|x) = 1 - p(y = 1|x)$, where $p(y = i|x)$ is the probability of instance x belonging to class i. So MAUC is also identical to its definition in the binary-class case.

The only exception is MFM. When $k = 2, MFM = \frac{1}{2}(F_1+F_2)$, where F_1 is identical to the definition of F-measure of the binary-class case in Eq. (4), where the minority class is the positive class, and F_2 is the F-measure treating the majority class as the positive class. Thus, F_1 favors the minority class while F_2 favors the majority class. When the change of F_2 dominates, or the changes of F_1 and F_2 are similar, the MFM value as the average of F_1 and F_2 will be misleading. For multi-class imbalanced data, MFM can be unsuitable when there are "multi-majority" classes [14], for example, a data set with class sizes $\{10, 1000, 1000\}$.

Table. II shows an example of comparing different evaluation measures. The minority class has 10 examples and the majority class has 100 examples. To correctly classify 3 more minority class examples, different methods sacrifice different numbers of majority class examples. G-mean favors the minority class most, followed by AvgAcc, then by F-measure, i.e., the F_1 value of MFM. MFM favors the minority class least, it is more close to accuracy than to F-measure. From this example we can see that, methods having higher G-mean and AvgAcc values can have quite lower MFM values. It should be cautious to use MFM as evaluation measure for multi-class imbalance learning methods, especially when there are "multi-majority" classes.

V. EXPERIMENTS

In the experiments, we compare the proposed ChunkCombine (abbr. as CC) method with 7 Boosting-based methods: (1) Ada: standard multi-class AdaBoost using AdaBoost.M algorithm. (2) OAda: over-sampling the minority classes to obtain balanced training set, then perform AdaBoost.M. (3) SMAda: over-sampling the minority classes using SMOTE to obtain balanced training set, then perform AdaBoost.M. (4) SMB: SMOTEBoosting [33]. (5) OVNC9 [14]: over-sampling + AdaBoosting.NC [14], with $\lambda = 9$. AdaBoosting.NC [42] is a multi-class AdaBoost method utilizing negative correlations. (6) UAda: under-sampling the majority classes to obtain balanced training set, then perform AdaBoost.M. (7) EE: EasyEnsemble.M [37], the number of AdaBoost classifiers is 4. All AdaBoost uses CART [43] as base learning method, and we assume it outputs discrete values. The iteration numbers of all AdaBoost classifiers are 40. Both SMOTE procedures in SMAda and SMB use $k = 5$. The number of data chunks of ChunkCombine is computed as follows: for D_i, if $n_i/n_1 = 1$, then $m_i = 1$, and if $n_i/n_1 > 1$, $m_i = [\log_2 \lfloor n_i/n_1 \rfloor \times 10/\log_2 100]$. We use the original implementations for OVNC9 and EE.

Eight UCI data sets are used with information shown in Table V. 10 times random splitting is performed with 50% and 50% as the training and test set, respectively. The normal splitting setting for the experiments of standard classification problems is 70% and 30% data as the training and test set, respectively. But for class-imbalance problems, especially

TABLE III. THE RESULTS OF AVGACC, G-MEAN AND AUC, WHERE BOLD FACE INDICATES THE BEST METHOD. THE AVERAGE RANKS ARE ALSO RECORDED. PAIRWISE TWO-TAILED t-TESTS AND SIGN TESTS BOTH WITH SIGNIFICANCE LEVEL .95 ARE PERFORMED. THE ROW OF "W-T-L" SHOWS THE WIN-TIE-LOSE COUNTS OF CHUNKCOMBINE VS. EACH METHOD IN COLUMN. THE SIGNIFICANT RESULTS ARE IN BOLD FACE.

AvgAcc	Ada	OAda	SMAda	SMB	OVNC9	UAda	EE	CC
abalone	.609 ± .034	.619 ± .030	**.648 ± .061**	.617 ± .040	.579 ± .049	.616 ± .036	.640 ± .065	.631 ± .055
balance	.580 ± .030	.553 ± .020	.572 ± .016	.599 ± .012	.547 ± .029	.569 ± .017	.599 ± .031	**.609 ± .027**
glass	.665 ± .070	.670 ± .054	**.700 ± .035**	.670 ± .077	.609 ± .065	.671 ± .063	.674 ± .042	**.700 ± .044**
Ecoli	.810 ± .035	.814 ± .027	.809 ± .020	.813 ± .028	.716 ± .051	.808 ± .035	.818 ± .025	**.828 ± .025**
segment	.993 ± .009	**.996 ± .003**	**.996 ± .004**	.730 ± .341	.995 ± .004	.994 ± .005	.992 ± .008	.992 ± .006
yeast	.713 ± .015	.715 ± .026	.719 ± .024	.712 ± .028	.699 ± .022	.715 ± .018	.727 ± .035	**.737 ± .030**
satimage1	.742 ± .017	.735 ± .020	.766 ± .028	.744 ± .035	.725 ± .054	.739 ± .017	.767 ± .027	**.776 ± .027**
satimage	.888 ± .005	.886 ± .006	.889 ± .007	.880 ± .007	.808 ± .007	.885 ± .005	.895 ± .006	**.899 ± .006**
avg.rank	5.38	4.50	2.88	5.25	7.38	5.38	2.63	**1.88**
w-t-l	5-3-0	3-4-1	4-3-1	4-4-0	**7-1-0**	4-4-0	2-6-0	
G-mean	Ada	OAda	SMAda	SMB	OVNC9	UAda	EE	CC
abalone	.211 ± .274	.259 ± .275	.490 ± .199	.298 ± .259	.208 ± .272	.261 ± .279	.606 ± .087	**.631 ± .079**
balance	.191 ± .208	.180 ± .158	.063 ± .133	.033 ± .103	.436 ± .066	.131 ± .171	**.596 ± .032**	.571 ± .028
glass	.426 ± .305	.427 ± .304	.632 ± .072	.517 ± .282	.557 ± .079	.484 ± .269	.633 ± .046	**.653 ± .045**
Ecoli	.800 ± .039	.804 ± .033	.799 ± .023	.801 ± .033	.696 ± .060	.799 ± .039	.810 ± .026	**.812 ± .028**
segment	.993 ± .010	**.996 ± .003**	**.996 ± .004**	.597 ± .513	.995 ± .004	.994 ± .005	.992 ± .008	.993 ± .006
yeast	.548 ± .069	.507 ± .193	.633 ± .064	.495 ± .187	.574 ± .060	.548 ± .072	**.706 ± .040**	.701 ± .036
satimage1	.473 ± .252	.287 ± .304	.672 ± .062	.441 ± .311	.598 ± .226	.406 ± .283	.754 ± .037	**.771 ± .034**
satimage	.878 ± .007	.879 ± .006	.881 ± .008	.864 ± .008	.799 ± .009	.875 ± .007	.891 ± .006	**.895 ± .006**
avg.rank	5.50	5.13	3.63	6.25	5.25	5.63	2.38	**1.75**
w-t-l	**6-2-0**	6-1-1	6-1-1	**6-2-0**	**7-1-0**	5-3-0	1-7-0	
MAUC	Ada	OAda	SMAda	SMB	OVNC9	UAda	EE	CC
abalone	.829 ± .033	.829 ± .028	.821 ± .041	.819 ± .035	.693 ± .041	**.835 ± .029**	.808 ± .040	.819 ± .293
balance	.767 ± .020	.746 ± .019	.753 ± .018	.770 ± .019	.720 ± .028	.752 ± .017	.772 ± .020	**.780 ± .012**
glass	.925 ± .016	.924 ± .015	**.938 ± .011**	.930 ± .013	.765 ± .039	.925 ± .016	.906 ± .012	.921 ± .009
Ecoli	.959 ± .008	.957 ± .008	.957 ± .012	.953 ± .012	.823 ± .032	.957 ± .010	.960 ± .010	**.963 ± .010**
segment	.995 ± .007	.997 ± .002	.997 ± .003	.798 ± .257	.997 ± .003	.997 ± .002	.998 ± .002	**1.000 ± .000**
yeast	.893 ± .018	.894 ± .020	.892 ± .012	.892 ± .014	.799 ± .015	.896 ± .015	.895 ± .014	**.898 ± .013**
satimage1	.945 ± .010	0.947 ± .015	.953 ± .011	.953 ± .011	.828 ± .034	.942 ± .019	.948 ± .009	**.955 ± .007**
satimage	.987 ± .000	.987 ± .000	.988 ± .000	.987 ± .001	.885 ± .004	.987 ± .001	.988 ± .000	**.989 ± .000**
avg.rank	4.25	4.25	3.38	4.63	7.38	3.75	3.63	**2.13**
w-t-l	5-3-0	4-4-0	4-3-1	3-4-1	**8-0-0**	5-2-1	4-4-0	

TABLE IV. THE RESULTS OF AVERAGE RANKS WHERE THE BEST METHODS ARE IN BOLD FACE. AND THE RESULTS OF t-TESTS AND SIGN TESTS. THE TABULAR SHOW THE WIN-TIE-LOSE COUNTS OF CHUNKCOMBINE VS. EACH METHOD IN COLUMN, WHERE THE SIGNIFICANT RESULTS ARE IN BOLD FACE.

avg.rank	Ada	OAda	SMAda	SMB	OVNC9	UAda	EE	CC
AvgAcc	5.38	4.50	2.88	5.25	7.38	5.38	2.63	**1.88**
G-mean	5.50	5.13	3.63	6.25	5.25	5.63	2.38	**1.75**
MAUC	4.25	4.25	3.38	4.63	7.38	3.75	3.63	**2.13**
avg.	5.04	4.63	3.30	5.38	6.67	4.92	2.88	**1.92**
w-t-l	Ada	OAda	SMAda	SMB	OVNC9	UAda	EE	
AvgAcc	5-3-0	3-4-1	4-3-1	4-4-0	**7-1-0**	4-4-0	2-6-0	
G-mean	**6-2-0**	6-1-1	6-1-1	**6-2-0**	**7-1-0**	5-3-0	1-7-0	
MAUC	5-3-0	4-4-0	4-3-1	3-4-1	**8-0-0**	5-2-1	4-4-0	
total	**16-8-0**	**13-9-2**	**14-7-3**	**13-10-1**	**22-2-0**	**14-9-1**	**7-17-0**	

when there are very few examples in the minority classes, such a splitting will result in too few minority class examples in the test set to obtain reliable estimation of a classifier's performance. In other words, the variance of the results of different runs will be very large. For example, the glass data set has just 9 examples in the smallest class, when using 70%-30% splitting, there are only 2 or 3 examples in the test set. Since about half of the data sets have "multi-majority" classes, such as abalone, balance, segment, yeast, we do not use MFM as evaluation measure. We record G-mean, AvgAcc and MAUC results in Table. III. The average ranks are also recorded. Pairwise two-tailed t-tests and sign tests both with significance level .95 are performed. The significant results are in bold face. The overall results of average ranks and the overall results of statistical tests are shown in Table. IV.

From the tables we can see that, no matter what evaluation measure is used, our proposed method CC (ChunkCombine) is always the best method since it has the lowest ranks on AvgAcc, G-mean and MAUC. Though CC is not necessarily significant better than any of the compared methods when a single evaluation measure is used, its performance on different

TABLE V. DATA SET INFORMATION, WHERE k IS THE NUMBER OF CLASSES, THE COLUMN n_i INDICATES EACH CLASS SIZE, IN THE COLUMN OF "ATT.", "N" AND "C" INDICATES NOMINAL AND CONTINUOUS ATTRIBUTE, RESPECTIVELY.

	k	n_i	att.	imbalance rates
abalone	3	14/126/391	1N7C	1:9:27.9
balance	3	49/288/288	4N	1:5.87:5.87
glass	6	9/13/17/29/70/76	9C	1:1.4:1.9:3.2:7.8:8.4
ecoli	5	20/35/52/77/143	7C	1:1.75:2.6:3.85:7.15
segment	3	20/165/330	19C	1:8.25:16.5
yeast	4	30/44/163/429	8C	1:1.46:5.4:14.3
satimage1	5	25/50/100/200/400	36C	1:2:4:8:16
satimage	6	626/703/707/ 1358/1508/1533	36C	1:1.12:1.13: 2.17:2.41:2.45

evaluation measures are very stable, such that statistical tests considering all the results of AvgAcc, G-mean and MAUC show that CC is significantly better than all the compared methods. We have discussed the properties of different evaluation measures in Section IV. Different measures favor the minority class with different degrees, and the methods having higher values of measure A can have lower values of measure B. For example, EE ranks lower than SMAda on AvgAcc and G-mean, but it ranks higher on MAUC. And almost all the other methods except CC have similar phenomenons.

EE is the second best method and it is also quite stable when different measures are used. The priority of CC over EE lies in that it explores more majority class information. The third best method is SMAda, then followed by OAda and UAda. All of them are better than Ada (standard AdaBoost). Both of the former methods are over-sampling methods. Since they use much more majority class examples than under-sampling, it is not surprise that they are better than UAda. But obviously, they are very time-consuming than UAda. However, the performances of UAda and OAda are very close. SMAda is better than OAda. This is because it can relieve the over-fitting problem of random over-sampling.

UAda's performance is also very close to Ada. This is because it ignores too much information contained in the majority classes. It is surprise that SMB is slightly worse than Ada. The reason maybe that it uses AdaBoost.M2 in which the base learner is required to output real values. While in AdaBoost.M the base learner is required to output discrete values. We have discussed before that when many majority class examples are ignored, even the AdaBoost classifier could be unstable.

OVNC9 is the worst method in general and CC is always significantly better than it. OVNC9 is similar to OAda, but it is always much worse than OAda. The difference mainly lies in that OVNC9 uses AdaBoost.NC and OAda uses AdaBoost.M to train multi-class AdaBoost.

It should be noted that, the results also show that AvgAcc, G-mean, and MAUC are appropriate evaluation measures for multi-class imbalance learning. In fact, we also use MFM as measure and the results show that Ada is the best method[1]. The results contradict the results of AvgAcc, G-mean and MAUC, which strongly suggests that MFM is unsuitable for multi-class imbalance learning in many cases.

We also recode the running time of each method in Table VI. The CPU is Intel Core2, 2.99GHz, and the memory is 2G. We can see that CC is the most time consuming one, followed by EE. The time complexity of CC is especially large when there are many classes, or the imbalance rate is large, this is one shortcoming of this method. SMB has larger time complexity than Ada, OAda, SMAda and UAda because it generates synthetic examples using SMOTE is each iteration of AdaBoost, which requires the calculation of nearest neighbours. Ada, OAda, SMAda and UAda have similar time complexity. OVNC9 has the least time complexity because it uses AdaBoost.NC instead of AdaBoost in the learning process.

[1]Due to page limit, we will show the results of MFM in a longer version later.

TABLE VI. RUNNING TIME

(seconds)	Ada	OAda	SMAda	SMB	OVNC9	UAda	EE	CC
abalone	11.9	12.4	13.2	32.7	1.0	11.8	44.8	364.3
balance	14.5	14.8	14.9	22.0	2.1	14.7	52.9	120.2
glass	4.9	5.1	5.2	12.7	0.7	5.0	17.9	145.5
ecoli	7.3	7.6	7.7	19.2	0.9	7.3	27.4	111.8
segment	0.7	0.3	0.3	1.0	0.8	0.3	1.3	12.1
yeast	14.9	15.9	16.5	40.0	1.5	15.0	55.3	253.1
satimage1	18.1	20.2	20.8	47.9	2.3	18.2	65.0	792.2
satimage	172.4	176.2	181.3	434.1	23.8	172.5	622.0	1272.0
avg.	30.6	31.6	32.5	76.2	4.1	30.6	110.8	383.9

VI. CONCLUSIONS

When under-sampling is used for multi-class imbalanced data, many majority class examples are ignored because there are often multiple majority classes, and the minority class often has few data. To utilize the information contained in the majority class examples ignored by under-sampling, we proposes a method ChunkCombine. For each majority class, it performs under-sampling multiple times to obtained non-overlapping data chunks, such that they contain the most information that a data sample of the same size can contain. Each data chunk has the same size as the minority class to achieve balance. Then every possible combination of the minority class and each data chunk from every majority class forms a balanced training set. ChunkCombine learns a set of multi-class AdaBoost classifiers from the different training sets derived from all the possible combinations. Finally, all the weak learners are combined by hard voting. Experimental results show ChunkCombine is better than many other popular methods for multi-class imbalanced data when average accuracy, G-mean and MAUC are used as evaluation measures.

Besides, we discuss different evaluation measures and suggest that, a multi-class F-measure Mean F-Measure (MFM) is unsuitable for multi-class imbalanced learning in many situations because it is not consistent with the standard F-measure in binary-class case and it is close to accuracy.

Efficiency is an important issue for multi-class classification, it is interesting to find ways to further improve ChunkCombine by avoiding learning from all possible combinations of the data chunks of every majority class. Reducing the efforts of learning the well separated classes maybe a good start.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their valuable comments. This work was supported by NSFC (61105046), SRFDP (Specialized Research Fund for the Doctoral Program of Higher Education, by Ministry of Education, 20110092120029), and Open Foundation of National Key Laboratory for Novel Software Technology of China (KFKT2011B01).

REFERENCES

[1] A. P. Bradley, "The use of the area under the ROC curve in the evaluation of machine learning algorithms," *Pattern Recognition*, vol. 30, no. 6, pp. 1145–1159, 1997.

[2] Q. Yang and X. Wu, "10 challenging problems in data mining research," *International Journal of Information Technology and Decision Making*, vol. 5, no. 4, pp. 597–604, 2006.

[3] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[4] W. Fan, S. J. Stolfo, J. Zhang, and P. K. Chan, "AdaCost: Misclassification cost-sensitive boosting," in *Proceedings of the 16th International Confernece on Machine Learning*, Bled, Slovenia, 1999, pp. 97–105.

[5] H. He and E. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[6] N. Japkowicz and S. Stephen, "The class imbalance problem: A systematic study," *Intelligent Data Analysis*, vol. 6, no. 5, pp. 429–449, 2002.

[7] X.-Y. Liu, J. Wu, and Z.-H. Zhou, "Exploratory undersampling for class-imbalance learning." *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 39, no. 2, pp. 539–550, 2009.

[8] M. Kubat and S. Matwin, "Addressing the curse of imbalanced training sets: One-sided selection," in *Proceedings of the 14th International Conference on Machine Learning*, Nashville, TN, 1997, pp. 179–186.

[9] G. M. Weiss, "Mining with rarity: A unifying framework," *ACM SIGKDD Explorations*, vol. 6, no. 1, pp. 7–19, 2004.

[10] A. C. Tan, D. Gilbert, and Y. Deville, "Multi-class protein fold classification using a new ensemble machine learning approach," in *Proceedings of the 14th International Conference on Genome Informatics*, Yokohama, Japan, 2003, pp. 206–217.

[11] Y. Sun, M. S. Kamel, and Y. Wang, "Boosting for learning multiple classes with imbalanced class distribution," in *Proceedings of the 6th International Conference on Data Mining*, 2006, pp. 592–602.

[12] Y. L. Murphey, H. Wang, G. Ou, and L. A. Feldkamp, "Oaho: an effective algorithm for multi-class learning from imbalanced data," in *Proceeding of the 2007 International Joint Conference onNeural Networks*. IEEE, 2007, pp. 406–411.

[13] T. R. Hoens, Q. Qian, N. Chawla, and Z.-H. Zhou, "Building decision trees for the multi-class imbalance problem," in *Proceedings of the 16th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 2012.

[14] S. Wang and X. Yao, "Multiclass imbalance problems: Analysis and potential solutions," *IEEE Transactions on Systems, Man, and Cybernetics, Part B – Cybernetics*, vol. 42, no. 4, pp. 1119–1130, 2012.

[15] A. Fernández, V. López, M. Galar, M. José del Jesus, and F. Herrera, "Analysing the classification of imbalanced data-sets with multiple classes: Binarization techniques and ad-hoc approaches," *Knowledge-Based Systems*, 2013.

[16] M. Lin, K. Tang, and X. Yao, "Dynamic sampling approach to training neural networks for multiclass imbalance classification," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 4, pp. 647–660, 2013.

[17] X.-Y. Liu and Z.-H. Zhou, "Learning multi-class imbalanced data with optimal dichotomy weights," in *Proceedings of the 13th International Conference on Data Mining*, 2013.

[18] P. Cao, D. Zhao, and O. Zaiane, "Measure optimized wrapper framework for multi-class imbalanced data learning: an empirical study," in *Proceedings of the 2014 International Joint Conference on Neural Networks (IJCNN'14)*, 2013.

[19] M. Kearns and L. G. Valiant, "Cryptographic limitations on learning boolean formulae and finite automata," in *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, Seattle, WA, 1989, pp. 433–444.

[20] Z.-H. Zhou, *Ensemble Methods: Foundations and Algorithms*. Boca Raton, FL: Chapman & Hall/CRC, 2012.

[21] S. Geman, E. Bienenstock, and R. Doursat, "Neural networks and the bias/variance dilemma," *Neural Computation*, vol. 4, no. 1, pp. 1–58, 1992.

[22] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[23] E. Bauer and R. Kohavi, "An empirical comparison of voting classification algorithms: Bagging, boosting, and variants," *Machine Learning*, vol. 36, no. 1-2, pp. 105–139, 1999.

[24] Z.-H. Zhou, J. Wu, and W. Tang, "Ensembling neural networks: Many could be better than all," *Artificial Intelligence*, vol. 137, no. 1-2, pp. 239–263, 2002.

[25] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[26] C. Ferri, J. Hernndez-Orallo, and R. Modroiu, "An experimental comparison of performance measures for classification," *Pattern Recognation Letters*, vol. 30, pp. 27–38, 2009.

[27] D. J. Hand and R. J. Till, "A simple generalisation of the area under the roc curve for multiple class classification problems," *Machine Learning*, vol. 45, no. 2, pp. 171–186, 2001.

[28] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*. Addison Wesley, 1999.

[29] Z.-H. Zhou and X.-Y. Liu, "Training cost-sensitive neural networks with methods addressing the class imbalance problem," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 1, pp. 63–77, 2006.

[30] ——, "On multi-class cost-sensitive learning," *Computational Intelligence*, vol. 26, no. 3, pp. 232–257, 2010.

[31] D. Tax, "One-class classification: Concept-learning in the absence of counter-examples," Ph.D. dissertation, Technical University of Delft, 2001.

[32] 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 *Proceedings of the 4th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, 1998, pp. 164–168.

[33] N. V. Chawla, A. Lazarevic, L. O. Hall, and K. W. Bowyer, "SMOTEBoost: Improving prediction of the minority class in boosting," in *Proceedings of the 7th European Conference on Principles and Practice of Knowledge Discovery in Databases*, Cavtat-Dubrovnik, Croatia, 2003, pp. 107–119.

[34] C. Seiffert, T. Khoshgoftaar, J. V. Hulse, and A. Napolitano, "RUSBoost: A hybrid approach to alleviating class imbalance," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 40, no. 1, pp. 185–197, 2010.

[35] P. Viola and M. Jones, "Fast and robust classification using asymmetric AdaBoost and a detector cascade," in *Advances in Neural Information Processing Systems 14*, 2002, pp. 1311–1318.

[36] X.-Y. Liu and Z.-H. Zhou, *Imbalanced Learning: Foundations, Algorithms, and Applications*. John Wiley & Sons, Inc., 2013, ch. Ensemble methods for class imbalance learning, pp. 61–82.

[37] Q.-Q. Li and X.-Y. Liu, "Easyensemble.m for multiclass imbalance problem," *Pattern Recognition and Artificial Intelligence*, accepted.

[38] Y. Freund and R. Schapire, "A desicion-theoretic generalization of on-line learning and an application to boosting," in *Proceedings of the Computational learning theory*, 1995, pp. 23–37.

[39] J. Wu, S. C. Brubaker, M. D. Mullin, and J. M. Rehg, "Fast asymmetric learning for cascade face detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, pp. 369–382, 2008.

[40] G. I. Webb, "MultiBoosting: A technique for combining boosting and wagging," *Machine Learning*, vol. 40, pp. 159–196, 2000.

[41] Y. Yu, Z.-H. Zhou, and K. M. Ting., "Cocktail ensemble for regression," in *Proceedings of the 7th IEEE International Conference on Data Mining*, Omeha, NE, 2007, pp. 721–726.

[42] S. Wang, H. Chen, and X. Yao, "Negative correlation learning for classification ensembles," in *Proceedings of 2010 International Joint Conference on Neural Networks*, 2010, pp. 1–8.

[43] L. Breiman, J. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*. CRC Press, 1984.