Pattern Recognition 77 (2018) 160–172

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

A regularized ensemble framework of deep learning for cancer detection from multi-class, imbalanced training data

Xiaohui Yuan[a,b], Lijun Xie[c,*], Mohamed Abouelenien[b]

[a] College of Information Engineering, China University of Geosciences, Wuhan, China
[b] Department of Computer Science and Engineering, University of North Texas, Denton, TX, USA
[c] Second Affiliated Hospital, School of Medicine, Zhejiang University, Hangzhou, China

ARTICLE INFO

Article history:
Received 30 September 2017
Revised 14 November 2017
Accepted 17 December 2017
Available online 19 December 2017

Keywords:
Ensemble
Deep learning
Imbalanced data
Cancer detection

ABSTRACT

In medical diagnosis, e.g. bowel cancer detection, a large number of examples of normal cases exists with a much smaller number of positive cases. Such data imbalance usually complicates the learning process, especially for the classes with fewer representative examples, and results in miss detection. In this article, we introduce a regularized ensemble framework of deep learning to address the imbalanced, multi-class learning problems. Our method employs regularization that accommodates multi-class data sets and automatically determines the error bound. The regularization penalizes the classifier when it misclassifies examples that were correctly classified in the previous learning phase. Experiments are conducted using capsule endoscopy videos of bowel cancer symptoms and synthetic data sets with moderate to high imbalance ratios. The results demonstrate the superior performance of our method compared to several state-of-the-art algorithms for imbalanced, multi-class classification problems. More importantly, the sensitivity gain of the minority classes is accompanied by the improvement of the overall accuracy for all classes. With regularization, a diverse group of classifiers is created and the maximum accuracy improvement is at 24.7%. The reduction in computational cost is also noticeable and as the volume of training data increase, the gain of efficiency by our method becomes more significant.

1. Introduction

In many real-world applications, data annotation is expensive and the target of interest occurs much less frequently compared to the normal cases. In bowel cancer diagnosis, capsule endoscopy has been used as a screening method, which provides over 55,000 frames in a video. The majority of the frames in a video depict normal tissues and the cancer symptom is under-represented in the training frames. Such uneven distribution creates a difficult learning process for finding unbiased decision boundaries. In addition, multi-class problems complicate the learning process. Many methods have been developed to handle the learning from imbalanced data, which leverage One vs. One (OvO) or One vs. All (OvA) strategies. These conversions mostly result in degraded performance and elongated training time as the number of class increases.

The existing methods can be classified as data-level approaches or algorithmic-level approaches [1–3]. In data level approaches, undersampling or oversampling is applied to balance the minority and majority classes. The oversampling process differs in the way how synthetic examples are created. Some techniques randomly create synthetic examples while others create synthetic examples based on density distribution [4] or distance to the decision boundary [5]. Algorithmic-level approaches such as cost-sensitive learning methods assign higher costs to the minority class [6].

Hybrid methods were developed by integrating both sampling and algorithmic approaches to handle the imbalanced data sets such as boosting methods. AdaBoost [7] was originally introduced to sequentially train an ensemble of classifiers to achieve improved accuracy through an error minimization function. With the success of this method, AdaBoost was extended to multi-class classification.

The indirect conversion of AdaBoost transforms the multi-class into multiple binary classifications using the binarization methods such as AdaBoost.M2 [7] and AdaBoost.MH [8]. These methods require extended training time with many training iterations and the improvement of accuracy is limited to a large number of classes. The direct conversion applies the boosting method to the multi-class data sets by changing the loss function. AdaBoost.M1 [7] was introduced to train multi-class data sets in which the error bound is too strict compared to random guessing error for multi-class problems. Stage-wise Additive Modeling using Multi-class Exponential (SAMME) loss function [9] was developed to ease the error bound of AdaBoost.M1 by transforming it to that of random

* Corresponding author.
E-mail address: xyuan@cse.unt.edu (L. Xie).

https://doi.org/10.1016/j.patcog.2017.12.017
0031-3203/

guessing of C classes, i.e., $\frac{C-1}{C}$. Mukherjee and Schapire [10] developed a theoretical approach to identify the optimal requirements of the trained weak classifiers and introduced a general framework for multi-class boosting. Saberian and Vasconcelos [11] introduced two multi-class boosting algorithms using multi-dimensional codewords and predictors based on coordinate and gradient descents. The algorithms vary in the types of weak classifiers they support. However, these more recent multi-class boosting methods did not consider the deterioration in the classification performance when trained on multi-class imbalanced data sets.

Ensemble methods in the deep learning framework have been developed to address the imbalance problem [12,13]. Some algorithms either applied sequential oversampling [14] or undersampling [15] within the framework of AdaBoost to balance the classes. Other boosting algorithms modified the updating rule to assign higher weights for misclassified minority instances [16]. Recently, boosting methods have been extended to address the more severe case of multi-class imbalance in certain applications, where several minority and majority classes are present [17,18].

In this article, we introduce a regularized ensemble framework of deep learning to address the problem of learning from imbalanced, multi-class data set for cancer detection. The new algorithm incorporates a regularization parameter that accommodates imbalanced multi-class data sets and automatically regulates the error bound of each classifier in accordance with its performance. The parameter penalizes the classifier when it misclassifies examples that were correctly classified in the previous iteration. This strategy aims at reducing volume-induced bias by pushing the classifier to preserve correctly classified minority examples. Additionally, the algorithm applies a weighted stratified sampling technique where each class represents a stratum. The sampling procedure is performed on each class separately based on weighted data distribution to balance the classes and avoid bias towards certain classes.

The rest of this article is organized as follows. Section 2 presents the existing methods that deal with learning from imbalanced training data. Section 3 describes our proposed ensemble framework of deep learning. Section 4 presents the experimental results and discussion. Section 5 concludes the paper with a summary.

2. Related work

The imbalance problem is defined based on the ratio of the sizes of different classes in a given data set [19]. Algorithms are developed to learn from imbalanced data sets, which aim to suppress the bias towards the majority class. Kubat and Matwin [19] selected a number of majority examples close to the boundary using one-sided selection to handle the imbalance problem. Chawla developed SMOTE [14] that randomly creates synthetic minority examples to balance the minority and majority classes. Han et al. proposed Borderline-SMOTE [5] that extends the SMOTE idea to oversample the minority examples close to the decision boundary. Barua et al. [20] developed Majority Weighted Minority Oversampling Technique (MWMOTE), to efficiently handle imbalanced data by specifying weights for selected hard-to-classify minority examples based on the Euclidean distance from the nearest majority example. A clustering method is then used to generate synthetic minority examples based on their respective weights. Wang et al. [4] introduced an oversampling method that is based on data density. The technique adaptively creates a different number of examples around each minority example based on its difficulty. Cao et al. [21] introduced a structure-preserving oversampling method combined with interpolation-based oversampling to create synthetic minority examples using multivariate Gaussian distribution. This method expands the minority examples in the empty area of the example space by keeping the covariance structure of the minority class and creating protective variances in the Eigen dimensions. Lopez et al. [22] conducted a comparative study of the data level approaches such as SMOTE with algorithmic level approaches that assign greater weights to the minority examples. Additionally, the study compared a hybrid approach that combines both methods. The comparison concluded that both techniques are equivalent in effectiveness while the hybrid approach has better performance in some cases.

In addition to creating synthetic examples for the minority class, selecting a subset of examples from the majority class to balance the training data set is also being practiced. Yen and Lee [23] introduced a cluster-based undersampling approach using back-propagation neural networks and investigated the effect of undersampling on the class distribution. Batuwita and Palade [24] proposed a method that resamples data by first selecting the most informative examples using Support Vector Machines (SVM) on the original data, then using these examples to re-sample the data and balance the classes. Zhou [25] studied the effect of different oversampling and undersampling methods for bankruptcy prediction. The experiments concluded that proper sampling method depends on the number of examples in the minority class.

Besides sampling in the data space, feature space and sophisticated methods are explored to aid balancing the data set. Piras and Giacinto [26] utilized the nearest neighbor paradigm to artificially create examples in the feature space of the minority class according to its local distribution. Wasikowski and Chen [27] used feature selection to reduce the dimensionality of imbalanced data sets as well as to gain higher accuracy for the minority class. Drown et al. [28] developed a genetic algorithm-based data sampling method to improve the software quality modeling and address the imbalance problem in high-assurance systems. Garcia et al. [29] used evolutionary algorithms to propose a method that handles the imbalance by storing objects in the Euclidean space. New examples are classified by measuring their distances to the closest generalized exemplary. The algorithm selects the best-generalized exemplars for optimization. Yu et al. [30] applied the ant colony optimization algorithm to exclude majority examples that are less informative to balance the classes.

Boosting methods became popular to deal with imbalanced data sets in conjunction with many aforementioned data balancing strategies. Karakoulas and Taylor [31] proposed AdaUBoost where the weights updating rule and the loss function were modified to assign higher weights to minority examples within a boosting framework. Chawla et al. [14] integrated SMOTE with the boosting procedure which iteratively trains the balanced data after adding the randomly created synthetic examples. Chen et al. [32] introduced RAMOBoost, an oversampling procedure for the minority examples. The technique ranks the minority examples at each training iteration based on a sampling probability distribution. Seiffert et al. [15] alleviated class imbalance using random undersampling in RUSBoost. E-Adsampling [16] algorithm created synthetic examples and updated the example weight to improve the accuracy of imbalanced data sets. Liu et al. [33] incorporated an ensemble of SVMs with oversampling and undersampling techniques combined to improve the accuracy of the minority class with highly skewed data. Saadi et al. [34] developed a method that employs SVM to balance the minority and majority classes for highly imbalanced data sets in the biomedical field. He et al. [35] proposed ADASYN, a method that utilizes the weights of minority examples to generate more synthetic examples towards harder-to-classify minority examples compared to easier examples. Yuan and Ma [36] proposed a sampling-reweighing strategy to tune AdaBoost. The method initially oversamples the data and applies AdaBoost followed by adjusting the weights using genetic algorithms.

Recently, attention is drawn to the more challenging case of learning from imbalanced, multi-class data. Murphey et al. [37] proposed OAHO algorithm using neural networks to learn from multi-class imbalanced data sets. Experiments conducted on highly imbalanced UCI data sets showed an improved performance of minority classes compared to the OvO and OvA classification methods. Ghanem et al. [38] developed Multi-IM that extends a probabilistic relational technique, PRMs-IM [39], by embedding the balancing method to multi-class data. Jeatrakul and Wong [40] developed One-Against-All with Data Balancing (OAA-DB) that combines undersampling technique using complementary neural networks with the oversampling technique SMOTE using OvA approach. Fernandez et al. [6] compared the classification performance of multi-class imbalanced data sets using binarization techniques such as OvO and OvA to the performance of preprocessing the data samples and cost-sensitive learning with ad hoc approaches. Codella et al. [12] combined deep learning method with established machine learning approaches, creating ensembles of methods for segmenting skin lesions, as well as analyzing the detected area and surrounding tissue for melanoma detection. Xiao et al. [13] applied deep learning to an ensemble that incorporates multiple different machine learning models. Data were selected by differential gene expression analysis and a deep learning method was employed to ensemble the outputs of the classifiers. Qi et al. [41] integrated Adaboost with deep support vector machine. Adaboost is applied to select SVMs with the minimal error rate and the highest diversity. By stacking SVMs into layers, the method acquires a new set of deep features. The training data represented by these new features is regarded as the input for a SVM classifier. Rasti et al. [42] employed a mixture ensemble of convolutional neural networks for breast detection. Each convolutional neural network is a modular and image-based ensemble, which stochastically partition the image space through simultaneous and competitive learning.

Clearly, how to balance training data is a key issue in multi-class problems. Bae et al. [43] proposed a mix-ratio sampling method that uses an SVM to determine oversampling sizes for different minority classes. Navarro et al. [44] proposed a dynamic oversampling procedure in a memetic algorithm that uses neural networks. The method re-samples data in two steps. First, the examples of the minority classes are oversampled to partially balance the classes. Second, the memetic algorithm is applied to oversample the data and generate new patterns for the class with the least sensitivity. Tong et al. [45] proposed an analytical method to determine the re-sampling scheme by utilizing response surface and design of experiments methods. Areibi and Tempelman [46] proposed a dynamic sampling framework that automatically tunes the training set distribution by combining sampling techniques such as SMOTE, random undersampling, and random oversampling methods. Yuan and Abouelenien [47] proposed a boosting framework using sampling with a constant error parameter to deal with imbalanced face recognition. Wang and Yao [17] studied the effect of multi-minority and multi-majority classes on the learning process. The paper concluded that multi-majority classes pose increased harm to the learning process. The study additionally explored AdaBoost.Nc [48] with imbalanced, multi-class data sets. The method used a negative correlation learning algorithm that utilizes an ambiguity term to add explicit diversity.

3. Regularized ensemble framework of deep learning

In AdaBoost.M1 [7], the weak classifier is assigned a weight based on evaluation, which is computed as follows:

$$\alpha_t = \frac{1}{2}\log\left(\frac{1-\epsilon_t}{\epsilon_t}\right), \tag{1}$$

where ϵ_t is the error of the resulted classifier in training iteration t, which calculated as the weighted sum of the misclassified instances. This weight decides the contribution of the classifier to the prediction on an unseen instance. The weighted error is the controlling factor of α. To avoid negative weights, the following condition must hold

$$1-\epsilon_t > \epsilon_t.$$

Hence, the weighted error bound is

$$\epsilon_t < 0.5. \tag{2}$$

Following SAMME algorithm [9], the loss function is extended by adding a constant term to the weight function. This term is presented by the logarithm of the constant $C-1$, where C is the total number of classes in the data set. The weight to the classifier becomes

$$\alpha_t = \frac{1}{2}\log\left(\frac{1-\epsilon_t}{\epsilon_t}\right) + \frac{1}{2}\log(C-1). \tag{3}$$

Following the same rationale as AdaBoost.M1, the following relation must hold

$$(1-\epsilon_t)(C-1) > \epsilon_t.$$

Hence, the weighted error bound becomes

$$\epsilon_t < \frac{C-1}{C}. \tag{4}$$

3.1. Unpredictability regularization

There are possibilities that some examples are misclassified after being correctly classified by the previous classifier. This can be attributed to the uneven class distribution. That is, a classifier can correctly classify most majority examples at the expense of minority examples that were correctly classified in the previous iteration. Hence, the decision boundary is adjusted in the directions that favor majority examples.

To address these issues and accommodate imbalanced multi-class data sets, we introduce a regularization parameter to the convex loss function in the classifier weight calculation. The parameter penalizes the weight of the classifier if it misclassifies examples that were correctly classified in the previous iteration. The magnitude of the penalty is determined by the weights of these misclassified examples and the regularization varies according to the evaluation of each classifier. Our target is to keep the correct classification of the minority examples, push the decision boundary towards the minority classes, and avoid the bias introduced by the larger number of majority examples.

After each training iteration, the classifier is evaluated and the weight function is calculated according to Eq. (3). The regularization parameter δ is initialized to 1. We start with the weight function of SAMME in the first iteration and develop our parameter to address the multi-class imbalance problem from there.

$$\alpha_t = \frac{1}{2}\log\left(\frac{1-\epsilon_t}{\epsilon_t}\right) + \frac{1}{2}\log(\delta_t(C-1)). \tag{5}$$

The loss function adjusts the examples' weights to increase weights of misclassified examples and reduce weights of correctly classified ones as

$$w_t = \begin{cases} w_{t-1}(i)e^{-\alpha_t}, & \forall x_i,\ f_t(x_i) = y_i \\ w_{t-1}(i)e^{\alpha_t}, & \forall x_i,\ f_t(x_i) \neq y_i. \end{cases}$$

The weights are then normalized.

In a boosting process, if an example is repeatedly misclassified, it is considered a hard example and additional learners need to learn from it with higher priority. That is, the base learner that consistently misclassifies an example of a few examples is

penalized by adjusting its weight. In the following iterations, the weighted error is decomposed into two parts, which differentiate the examples that are consistently misclassified from those that are correctly classified by the previous classifiers but misclassified in the current training round.

$$\epsilon_t = \epsilon_c + \epsilon_m. \tag{6}$$

The first term ϵ_c is the sum of weighed error of the examples that are misclassified by the current classifier and are correctly classified by the previous one. Such examples are named second-round-misclassified examples $X_c = \{x_i; f_t(x_i) \neq y_i \text{ and } f_{t-1}(x_i) = y_i\}$ and the error is computed as follows:

$$\begin{aligned}\epsilon_c &= \sum_{i \in X_c} w_{t-1}(i) \\ &= \sum_{i \in X_c} w_{t-2}(i) \left(\frac{\delta_{t-1}(C-1)(1-\epsilon_{t-1})}{\epsilon_{t-1}} \right)^{-\frac{1}{2}}.\end{aligned} \tag{7}$$

The second term in Eq. (6) ϵ_m is the sum of weighed error of the examples that are misclassified by two consecutive classifiers, i.e., $X_m = \{x_i; f_t(x_i) \neq y_i \text{ and } f_{t-1}(x_i) \neq y_i\}$. The error is computed as follows:

$$\begin{aligned}\epsilon_m &= \sum_{i \in X_m} w_{t-1}(i) \\ &= \sum_{i \in X_m} w_{t-2}(i) \left(\frac{\delta_{t-1}(C-1)(1-\epsilon_{t-1})}{\epsilon_{t-1}} \right)^{\frac{1}{2}}.\end{aligned} \tag{8}$$

The goal of the iterative boosting method is to adjust the decision boundary to correctly classify hard examples. However, alternately treating an example differently results in fluctuation in the weighted data distribution and, hence, unpredictability of classifier. The regularization term δ_t adjusts the classifier's weight to penalize the one that misclassifies the second-round-misclassified examples. It measures the difference in the weighted error if all currently misclassified examples were also misclassified in the previous iteration. In this case, to derive the explicit expression for δ_t, we assume that all currently misclassified examples are misclassified by the previous classifier, i.e., the exponent of the right-hand term in Eq. (7) is positive to increase the weights of the second-round-misclassified examples expressed by ϵ_c. Using this assumption of two consecutive classifiers, the maximum possible weighted error ϵ'_t is

$$\epsilon'_t = \sum_{i \in (X_c \cup X_m)} w_{t-2}(i) \left(\frac{\delta_{t-1}(C-1)(1-\epsilon_{t-1})}{\epsilon_{t-1}} \right)^{\frac{1}{2}}. \tag{9}$$

The examples weights are re-normalized accordingly. The actual weighted error is equivalent to the maximum possible weighted error ϵ'_t altered by the parameter $\delta^{\frac{1}{2}}$. The regularization provides a measurement for the deterioration caused from the second-round-misclassified examples. The value of the parameter is hence proportional to these examples' weights

$$\epsilon_t = \epsilon'_t \delta_t^{\frac{1}{2}}. \tag{10}$$

Since $\epsilon_t \leq \epsilon'_t$, the parameter is less than or equal to 1, i.e., $\delta \leq 1$.

By substituting Eq. (10) in Eq. (9), we can derive the explicit formula for the regularization term δ_t as follows:

$$\delta_t = \frac{\epsilon_t^2 \epsilon_{t-1}}{Z^2(1-\epsilon_{t-1})\delta_{t-1}(C-1)}. \tag{11}$$

where Z is the sum of examples' weights

$$Z = \sum_i w_{t-2}(i).$$

The classifier weight α is adjusted by integrating the logarithm of δ_t, which penalizes the classifiers that result in second-round-misclassification as shown in Eq. (5). δ_t varies during each iteration based on the performance of the two consecutive classifiers.

In this scheme, the weighted error must satisfy

$$(1-\epsilon_t)\delta_t(C-1) > \epsilon_t.$$

Hence, the error bound for classifier t becomes

$$\epsilon_t < \frac{1}{1+\delta_t^{-1}(C-1)^{-1}}. \tag{12}$$

3.2. Regularized ensemble framework

Given a training data set $D = \{(x_1, y_1), \ldots, (x_M, y_M)\}$, where an example $x_i \in \mathbb{R}^N$ and its label $y_i \in \{1, \ldots, C\}$, C is the number of classes in the data set and M is the total number of examples in D. In our data sampling, we follow the spirit of stratified sampling and treat each class as a stratum. Clearly, not all classes need to be resampled and the class sampling rate is inversely proportional to its imbalance rate. The sampling of a stratum is random following the data distribution. That is, the weight of each example represents its probability of being included in the training set. The selection process deals separately with each class to recover the balance and ensure that all the classes are represented in the sampled data. This stratified sampling is different from random undersampling in that the hard examples are more likely to be kept in the training set as their weights are greater. Let $|c_i|$ denote the size of class c_i. The smallest minority class is $|c^*| = min_{i \in \{1,2,\ldots,C\}}(c_i)$. For each class in the data set D, stratified sampling selects a subset of examples, denoted with d_i, from each class c_i such that number of selected examples from a majority class equals to the size of the smallest minority class, i.e., $|d_i| = |c^*|$. The balanced training set becomes collection of sampled majority classes and the minority classes:

$$D_t = \cup_i d_i \subset D. \tag{13}$$

Algorithm 1 presents our method. The weight of examples is initialized to be equal at $\frac{1}{M}$ and the regularization parameter δ is initialized with 1. Function $\mathbb{1}_0^1[\cdot]$ is an indicator function that returns 1 if true and 0 otherwise in order to calculate the weighted error, and function $\mathbb{1}_{-1}^1[\cdot]$ denotes an indicator function that returns 1 if true and -1 otherwise in order to increase or decrease the weights of the examples.

With a balanced training set D_t, a classifier f_t is trained that minimizes the weighted error of D_t. Since data distribution plays a key role in both error minimization (as shown in Eq. (14)) and data sampling, we need to have a complete view of the classifier's empirical performance. Therefore, despite that f_t is trained with D_t, which is a subset of D, the classifier is evaluated using the entire data set such that the weight of each example is updated, i.e.,

$$\epsilon_t = \sum_{i \in D} w_{t-1}(i) \mathbb{1}_0^1[y_i \neq f_t(x_i)].$$

Except in the first iteration, the regularization parameter δ is computed according to the errors of two consecutive classifiers, which is used in both calculation of error bound and classifier weight.

4. Experimental results and discussion

4.1. Data sets and experiment setup

In our experiments, we used both synthetic and real-world data sets including numerical data and imagery. Our Capsule Endoscopy (CE) video set consists of data from eight different patients. The imbalance ratio of the CE data set is much higher than the other

Algorithm 1 Regularized ensemble framework.

Input: $D = \{(x_1, y_1), \ldots, (x_M, y_M)\}$
Output: f_t and α_t

Training

Initialize example weight $w_0(i) = \frac{1}{M}$ and $\delta_1 = 1$
for $t = 1, \ldots, T$ **do**
  Select a subset of examples D_t using stratified sampling and data distribution w_t.
  Form a training set following Eq. (13).
  Train a classifier f_t with D_t

$$f_t \Leftarrow \arg\min \sum_{i \in D_t} w_{t-1}(i) \mathbb{1}_0^1[y_i \neq f_t(x_i)] \tag{14}$$

  if $t > 1$ **then**
    Calculate δ_t following Eq. (11).
  end if
  if $\epsilon_t > \frac{1}{1+\delta_t^{-1}(C-1)^{-1}}$ **then**
    return $\alpha_t \Leftarrow 0$
  else
    Compute weight α_t for classifier f_t

$$\alpha_t \Leftarrow \frac{1}{2}\log(\frac{1-\epsilon_t}{\epsilon_t}) + \frac{1}{2}\log(\delta_t(C-1)) \tag{15}$$

    Calculate example weight:

$$w_t(i) \Leftarrow w_{t-1}(i) e^{(\alpha_t \mathbb{1}_{-1}^1[y_i \neq f_t(x_i)])}$$

    Normalize $w_t(i)$:

$$w_t(i) \Leftarrow \frac{w_t(i)}{\sum_{j=1}^{M} w_t(j)}$$

  end if
end for

Testing

Integrate classifiers f_t with weighted sum:

$$F(x) \Leftarrow \arg\max_y \sum_{t=1}^{T} \alpha_t f_t(x)$$

data sets, which is common in real-world applications. To compare with the state-of-the-art methods, two multi-modal Gaussian data sets, namely SYN1 and SYN2, are generated by randomly varying the means and variances, which changes the overlaps among classes. Classes in SYN2 are closer to each other on average compared to that of the classes in SYN1. Both synthetic data sets are two dimensional. Other real-world data sets include four from UCI repository [49] Image Segmentation (Image), Letter Recognition (Letter), Pen-Based Recognition of Handwritten Digits (Pen), and Statlog Landsat Satellite (Statlog) and two popular face data sets (AT&T and AR). Images in the AR data set were converted into grayscale and manually cropped and aligned. The cropped image size is 120×110. Face images were preprocessed using principal component analysis for dimensionality reduction before the training. Test images were projected on the Eigen-space before testing. In addition, we used binary-class data sets, among which two are popular synthetic data sets Circle-Disk (CD) and banana. The Circle-Disk set is formed of a disk (one class) surrounded by a circle (the other class), and the banana set consists of two banana-shaped classes protruding into each other. Including binary cases is to demonstrate the efficacy of the methods, since it essentially is a special case of the multi-class classification. Additional details of the data sets are summarized in Table 1. Note that image data sets are high dimensional data sets.

Table 1
Data sets and properties.

Data sets	No. of classes	Set size	No. of features	Min. size	Maj. size	Imb. ratio
CE	2	4200	36,864	25	2075	1:83
SYN1	50	5000	2	5	50	1:10
SYN2	50	5000	2	5	50	1:10
Image	7	2310	18	15	165	1:11
Letter	26	18,200	16	50	350	1:7
Pen	10	10,540	16	31	527	1:17
Statlog	6	3744	36	24	312	1:13
AT&T	40	400	10,304	2	9	1:4.5
AR	50	550	13,200	2	10	1:5
CD	2	4000	2	50	1000	1:20
Banana	2	4000	2	50	1000	1:20

Without loss of generality, each data set was divided into a minority group and a majority group. The size of classes in each group is the same except Image data set which has an odd number of classes. In our experiments, two-fold cross-validation was conducted and classes in both groups are then switched. That is, each class serves as the minority class and the majority class. Limited by the number of examples in each class in the AT&T and AR data sets, we used leave-one-out cross-validation instead. The two classes of the CE data were not switched since the data set is naturally imbalanced and the minority class (diseased images) has very few examples compared to the majority class (normal images).

In constructing ensembles, we used both decision trees and deep neural network as base classifiers. Early pruningc was employed in the decision tree implementation to avoid overfitting. Each ensemble is trained with 100 iterations, i.e., 100 classifiers are created. The results of our proposed method are compared against the state-of-the-art algorithms for multi-class, imbalanced classification including AdaBoost.M1, SAMME, RUSBoost, and SMOTEBoost. To extend RUSBoost for multi-class problems, we followed AdaBoost.M1 strategy. Our preliminary experiments of SMOTEBoost using AdaBoost.M1 and AdaBoost.M2 for multi-class problem showed that results with AdaBoost.M1 had higher sensitivity, accuracy, and efficiency, and was hence adopted in our multi-class extension for RUSBoost and SMOTEBoost.

4.2. Classification of the minority classes

Analysis of classification performance of the minority classes in a multi-class problem is complicated because the impact of imbalance to the discriminant among classes is usually heterogeneous. For instance, if a large margin exists between two classes the impact of imbalance is much less than the case where two classes are heavily overlapped. In addition, it is not trivial to characterize the class layout especially for high dimensional scenarios. Hence, we report the statistical performance of classification using average sensitivity $\bar{\epsilon}$ and accuracy $\bar{\rho}$ as follows:

$$\bar{\epsilon} = \sum_i \frac{\sum_{k \in c_i} \mathbb{1}_0^1[y_k = F(x_k)]}{|c_i|} \tag{16}$$

$$\bar{\rho} = \sum_j \frac{\sum_{k \in c_j} \mathbb{1}_0^1[y_k = F(x_k)]}{|c_j|} \tag{17}$$

where $|c_l|$ denotes the size of class l, c_i denotes the minority classes, and c_j denotes any class in the data set. Again, $\mathbb{1}_0^1[\cdot]$ is an indicator function that gives one if the condition is true and zero otherwise.

Table 2
The average sensitivity (in percentage %) of minority classes and accuracy in multi-class problems. The numbers in parenthesis are the standard deviation. The underlined results are the best among all learner and method combinations. The dash lines denote unsuccessful training. The methods included are AdaBoost.M1 (Ada), SAMME, RUSBoost (RUS), SMOTEBoost (SMT), and our method (REFDL).

Data	Decision trees					Deep networks				
sets	Ada	SAMME	RUS	SMT	REFDL	Ada	SAMME	RUS	SMT	REFDL
Sensitivity										
SYN1	20.6	20.3	53.94	55.6	**58.3**	1.7	0.9	55.0	63.5	**63.6**
	(5.0)	(4.9)	(11.3)	(5.8)	(12.7)	(3.4)	(1.7)	(9.9)	(9.9)	(3.2)
SYN2	13.2	13.1	14.9	39.7	**43.1**	1.04	0.9	–	12.3	**45.6**
	(4.8)	(4.8)	(12.8)	(2.0)	(6.1)	(1.4)	(1.1)	–	(22.0)	(2.7)
Image	38.1	38.1	78.1	71.3	**82.9**	80.8	80.7	**91.9**	88.4	88.5
	(7.4)	(7.0)	(3.9)	(3.9)	(3.7)	(6.9)	(5.7)	(2.4)	(3.6)	(2.3)
Letter	18.9	19.0	–	35.4	**59.5**	2.4	3.4	61.1	63.8	**67.7**
	(7.5)	(7.5)	–	(4.1)	(5.8)	(1.8)	(3.8)	(3.0)	(2.4)	(2.1)
Pen	43.4	43.1	81.0	74.7	**86.6**	68.2	76.9	93.0	89.0	**94.2**
	(8.8)	(8.3)	(3.1)	(4.7)	(3.1)	(11.2)	(1.4)	(2.8)	(6.0)	(2.0)
Statlog	45.8	46.0	72.2	64.2	**73.5**	61.6	58.8	75.1	72.5	**75.3**
	(9.4)	(9.5)	(10.9)	(12.8)	(14.9)	(10.9)	(9.3)	(14.7)	(12.2)	(14.2)
AT&T	18.7	18.6	–	**42.5**	40	1.2	0.5	–	37.5	**60.0**
	(8.1)	(8.0)	–	(13.3)	(10.9)	(2.9)	(3.1)	–	(15.5)	(16.3)
AR	20.0	20.5	–	34.0	**46.0**	1.0	0.0	–	2.0	**29.0**
	(9.6)	(9.4)	–	(11.6)	(13.5)	(2.0)	(0.0)	–	(3.5)	(10.0)
Accuracy										
SYN1	47.8	47.9	58.9	**61.6**	57.3	41.4	41.6	61.5	**64.0**	63.0
	(3.4)	(3.4)	(4.1)	(0.7)	(3.8)	(2.0)	(2.0)	(3.5)	(1.7)	(1.1)
SYN2	37.5	37.8	19.3	**43.5**	35.7	35.5	35.5	–	12.5	**40.8**
	(2.8)	(2.6)	(12.1)	(1.2)	(5.9)	(1.2)	(0.7)	–	(21.1)	(0.9)
Image	61.9	62.0	76.3	74.2	**83.3**	88.3	87.8	89.1	89.5	**90.8**
	(8.4)	(8.5)	(5.2)	(6.6)	(3.1)	(4.1)	(3.6)	(3.6)	(3.2)	(1.6)
Letter	41.2	41.1	–	48.3	**55.9**	40.8	41.4	**63.3**	63.2	57.6
	(3.5)	(3.6)	–	(2.7)	(5.9)	(1.6)	(2.6)	(1.7)	(1.4)	(1.7)
Pen	66.3	66.5	83.5	80.8	**88.1**	83.3	87.7	94.2	92.6	**96.5**
Pen	(3.6)	(3.5)	(0.9)	(2.6)	(1.9)	(5.4)	(0.8)	(1.6)	(3.4)	(1.4)
Statlog	63.8	63.8	75.2	72.1	**76.1 (1.0)**	75.8	74.1	**80.0**	79.8	78.5
	(2.8)	(2.8)	(1.4)	(3.8)	(1.0)	(4.5)	(5.2)	(2.3)	(2.6)	(4.4)
AT&T	35.6	35.5	–	**40**	35.8	11.8	40.2	–	**56.2**	48.7
	(8.2)	(8.4)	–	(8.6)	(9.0)	(15.2)	(6.2)	–	(14.9)	(11.0)
AR	28.5	28.5	–	**35.5**	29.0	8.0	19.0	–	2.0	**22.0**
	(9.5)	(9.5)	–	(10.6)	(8.9)	(5.1)	(6.6)	–	(5.1)	(6.7)

Table 2 summarizes the average sensitivity and accuracy, and the standard deviation is reported in parenthesis. The best values achieved are highlighted with bold font face. In general, the sampling-based methods consistently exhibit greater performance in both sensitivity and overall accuracy. The average improvement is 2 to 3 folds with respect to both metrics. However, there are four cases that RUSBoost was unsuccessful to create an ensemble. The cause of these failures is rooted in the random under-sampling scheme and is also a result of joint factors including problem complexity and base learner. Further discussion will be given in the next section.

Our method achieves the best minority class sensitivity in most cases and is the second best in the rest cases. It is important to note that despite large average sensitivity is achieved by our method, its overall accuracy remains very competitive. Using decision trees as the base learner, our method yielded the best sensitivity and accuracy for four cases; using neural networks, it resulted in the best performance in terms of both metrics for three cases. In addition, the standard deviations of our method are small, which demonstrate the consistency of its performance. The underlined results in Table 2 highlight the best performance among all learner and method combinations. It is evidential that our method presents the superior sensitivity (7 out of 8 cases) and still achieves competitive accuracy (2 out of 8).

It is interesting to note that using neural networks as the base learner AdaBoost.M1 and SAMME resulted in extremely low sensitivity but relatively larger standard deviation. This is because there are a small number of cases that yield very low performance but the majority is relatively higher. That is the performance is skewed and the median is greater than the mean.

Although our focus is on the multi-class problem, we also conducted some experiments with binary classification problems. Our results are reported in Table 3. With binary problems, both RUSBoost and SMOTEBoost achieved much greater performance in contrast to AdaBoost.M1 and SAMME. Among three cases, SMOTEBoost yields the best sensitivity and accuracy in both synthetic data sets. It is appropriate to say that with small to moderate amount of training data SMOTEBoost is very competitive. However, with large data and dimensionality as in CE data set, SMOTEBoost failed training due to a large number of synthetic samples created. In handling large data sets, our method exhibited both robustness and the best performance. Although RUSBoost gives the highest sensitivity using the neural network, the result of our method is extremely close with a smaller standard deviation. It is plausible to say that our method is a competitive method in dealing with binary classification problems and has a great advantage in handling large data sets.

4.3. Effective classifiers in ensemble

An ensemble relies on the diversity of its classifiers to model the data distribution closely. To deal with multi-class problems our method, as well as many other state-of-the-art methods, introduces a lower error bound to the learning process. This ensures more training iterations to be conducted but also allows classifiers to have very low weights. When the weight of a classifier is close to zero, it essentially has little contribution to the final decision despite the time for training as well as the time taken to process a new instance. The number of effective classifiers (NEC) is hence an important property of an ensemble.

Table 3
The average sensitivity (in percentage %) of minority classes and accuracy in binary classification problems. The numbers in parenthesis are the standard deviation.

Data	Decision trees					Deep networks				
sets	Ada	SAMME	RUS	SMT	REFDL	Ada	SAMME	RUS	SMT	REFDL
Sensitivity										
CE	8.0	8.1	42.0	–	**50.0**	16.0	16.5	**82.8**	–	82.0
	(11.3)	(10.9)	(8.4)	–	(19.7)	(16.9)	(15.8)	(5.6)	–	(2.8)
CD	2.3	2.5	**56.0**	52.4	54.5	53.0	53.3	51.7	**94.8**	66.3
	(4.7)	(4.8)	(43.7)	(51.4)	(39.6)	(19.1)	(18.8)	(34.1)	(6.3)	(19.9)
Banana	16.9	16.9	54.3	**96**	53.9	51.6	52.0	81.1	**91.0**	69.6
	(13.1)	(13.2)	(13.3)	(2.7)	(15.9)	(5.7)	(6.0)	(3.5)	(1.1)	(3.7)
Accuracy										
CE	53.6	53.6	68.9	–	**72.7**	57.8	57.9	73.5	–	**73.8**
	(5.7)	(5.9)	(3.8)	–	(7.5)	(8.4)	(8.2)	(2.2)	–	(1.2)
CD	51.1	51.3	**72.5**	60.5	72.0	75.6	75.9	74.3	**91.4**	81.9
	(2.2)	(2.1)	(16.7)	(14.4)	(14.8)	(9.2)	(8.9)	(17.2)	(1.5)	(9.7)
Banana	58.4	58.4	75.2	71.9	**75.4**	75.6	74.9	88.6	**89.7**	84.0
	(6.5)	(6.5)	(5.5)	(2.5)	(6.1)	(2.7)	(2.6)	(1.4)	(0.9)	(1.8)

Table 4
Average number of effective classifiers.

Data	Decision trees					Deep networks				
set	Ada	SAMME	RUS	SMT	REFDL	Ada	SAMME	RUS	SMT	REFDL
CE	10.5	10	45.5	–	14	2.5	2	13.5	–	7.5
SYN1	1.75	1	13.5	2.5	76.75	10	26.5	7.5	9.5	68.25
SYN2	2.75	1.25	1	1.75	74.75	6.75	35.75	0	1	76.5
Image	1.5	2.5	24.5	1.5	83	7.5	9.75	17.25	12.25	30
Letter	1.75	2.25	0	2	80.5	7.5	28.5	8.25	7.75	63.75
Pen	2.25	2	23.25	2.75	81.25	9.75	20.25	15	12	31.75
Statlog	1.75	1	16.5	1.75	71.75	10	14.5	14.75	6	36.75
AT&T	1.4	1.4	0	1.35	54.65	1.6	37.8	0	3.8	79.85
AR	1.68	1.72	0	1.9	66	1	59.5	0	1	73.8
CD	1.25	1.5	27.75	2	3.5	6.5	7	22.75	6.75	25.75
Banana	1.5	1.5	15.25	1.75	9.25	5.75	6	22	6.75	13

Table 4 summarizes the average number of effective classifiers in the ensemble out of 100 training iterations. Recall that there is no value reported for the sensitivity or accuracy of some experiment cases. This is explainable with NEC values. For example, by cross-referencing Tables 2 and 4 it is easy to see that when NEC is zero the ensemble yields no results. When NEC is one, that is there is one classifier that has a non-zero weight, the ensemble reduces to a single classifier and its performance is very low in most cases.

On the other hand, large NEC values could be an indicator of overfitting as well, especially for the under-sampling based methods. For instance, in learning from CD data set RUSBoost resulted in an average of 22.75 effective classifiers using neural network; however, its sensitivity and accuracy are not proportional to its NEC. The possible cause of this inconsistency is that random under-sampling has great potential of missing key instances when a small number of examples are selected each time. Clearly, this random process diversifies the classifier, yet it generates a misrepresentation of the underlying model. Our proposed method, on the other hand, acts consistently in all cases. A large number of effective classifiers helps our method achieves the best or very competitive performance.

4.4. Training efficiency

The computational cost is a major consideration in the employment of ensemble learning algorithms. With the iterative training scheme, it usually takes longer time than any single classifier, which becomes an issue when dealing with large, high-dimensional data sets. Our programs are implemented with MATLAB in a 64-bit Windows system with Intel Core 2 Duo processor at 3 GHz. The system has 4GB memory and 6.6GB virtual memory.

Table 5 summarizes the average time used for training ensemble classifiers. The second column lists the data volume in the training phase, which is the product of the number of examples used for training and the number of features per example. The data sets are put in descending order according to the training data volume. The shortest time is highlighted in bold font face.

Since RUSBoost, SMOTEBoost and our proposed method require a data resampling step, intuitively they take a longer time to complete training. This is true when the problem is relatively simple; both Adaboost.M1 and SAMME took much less time to process the synthetic data sets using decision trees as the base classifier. However, as the problem complexity grows, especially as the training data volume increases, the advantage of the under-sampling based methods reveals. RUSBoost and our method took the least amount of time to process all eight real-world data sets. RUSBoost failed to achieve an effective ensemble for Letter, AT&T, and AR data sets. Hence the time used to generate RUSBoost ensemble is omitted. SMOTEBoost, on the other hand, suffers from processing the additional synthetic examples; the extended time cost is significant with high training data volume. In particular, when training with CE data SMOTEBoost was unsuccessful to complete due to the extremely large amount of data generated for training. As shown in Table 1, the imbalance ratio of CE is 1:83. So the SMOTEBoost method has to handle almost 80 times more data than any other methods, which causes it to fail given the limited amount of memory in our computer system.

Using neural network as the base classifier requires longer training time to train an ensemble. It is evident that our proposed method is very predictable in its superior efficiency among all methods. The training time consumed by RUSBoost and our method is close, although our method yielded the shortest average time for 8 cases among the 11 data sets. RUSBoost again failed

Table 5
Time (in second) used in training ensemble classifiers.

Data	Training	Decision trees					Deep networks				
sets	Volume	Ada	SAMME	RUS	SMT	REFDL	Ada	SAMME	RUS	SMT	REFDL
CE	77,414,400	1012	1012	29.2	–	**27.6**	10,892	10,902	314	–	**293**
Banana	2100	1.42	**1.41**	1.7	2.6	2.5	859	854	510	910	**426**
CD	2100	**1.21**	1.22	1.8	2.5	2.7	1029	1030	**424**	1072	475
SYN2	2750	50.6	50.6	**41.6**	88.7	47.4	3027	3159	–	4856	**1511**
SYN1	2750	38.4	**38.3**	38.7	50.2	41.8	3558	3595	1809	5136	**1679**
Image	11,340	6.7	6.5	**5.3**	13.7	6.3	1400	1032	**708**	1494	785
Statlog	36,288	12.3	12.3	**5.1**	28.9	7.2	1171	1196	**619**	1514	746
Pen	44,640	31.8	30.3	**12.4**	76.1	16.4	3142	2318	1283	4033	**993**
Letter	83,200	213	221	–	409	**108**	6587	6701	2108	11,373	**1718**
AT&T	2,266,880	27.2	27.1	–	551	**26.1**	917	936	–	1597	**727**
AR	3,960,000	62.2	62.8	–	1059	**40.6**	1198	1144	–	2421	**832**

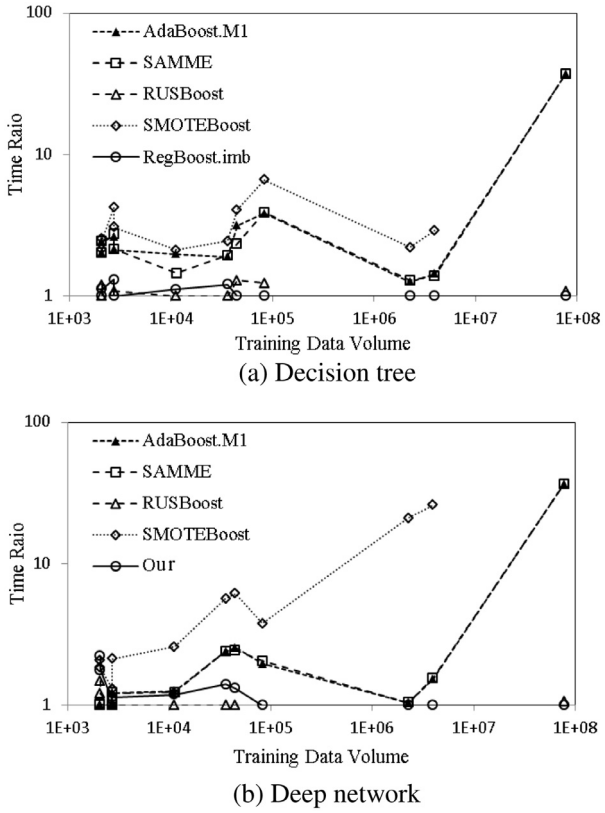

Fig. 1. Average relative time used in training. The discontinued or incomplete curves indicate failure of generating an ensemble.

to generate effective ensemble for three data sets. Compared to AdaBoost.M1 and SAMME methods, our proposed method reduces the training time by more than 50%. In particular, in the CE case our method used a fraction of the time needed for either AdaBoost.M1 or SAMME.

Fig. 1 illustrates scatter plots of the relative average time used in training with respect to the training data volume. The plots show the ratio of time used by each method with respect to the minimum time in each case. To reveal the difference, the vertical axis is in logarithm scale. Although the time is probably affected by the hardness of the problem and hence curves are not increasing functions, the relative computational cost among methods is clearly depicted where our method exhibits the best efficiency among all cases especially when dealing with a large amount of data as shown in the plot. Despite the absolute amount of time used with a different base classifier, this trend remains consistent.

4.5. Regularization parameter

In our method, the regularization parameter δ plays a vital role in the algorithm's capability and performance. In this section, we studied its properties as well as its behavior with respect to the performance to gain a deep insight into this strategy.

In general, an increase of δ between two consecutive training iterations indicates that the number of second-round-misclassified instances decreases. However, the amplitude of the change of δ is also determined by the example weights. That is, an "easy" example that has been correctly classified by several previous consecutive classifiers and is labeled incorrectly makes a small increment to the δ compared to those borderline examples that are repeatedly correctly and incorrectly classified. The weight associated with such an easy example is clearly less than the weight of a borderline example, which in turn limits its contribution to the change of δ. In addition, the employment of different training examples injects variations into the regularization parameter. It is hence anticipated that δ fluctuates during the training process, which is evidential from Fig. 2.

Fig. 2 illustrates the changes of regularization parameter with respect to the training iteration. The horizontal axis gives the iteration number (up to 100) and the vertical axis shows the δ values in the range of $(0, 1]$. The solid line and dash line depict δ with the decision tree and neural network as a base learner, respectively. Despite some disparities in fluctuation between two base learners, the overall behaviors mostly coincide.

Recall that δ is initialized with one in the first training iteration. Hence, its value tends to drop sharply at the very beginning because there are surely many examples that are misclassified. The weight of these classifiers, i.e., α, then relies mostly on the performance-based evaluation term $\log(1-\epsilon_t)/\epsilon_t$ as shown in Eq. (3). As training continues, δ tends to increase, which allows more contribution from the multi-class regularization term $\log(C-1)$ to the classifier weight and shows that the method was successful in reducing the number of second-round-misclassified examples.

Since we successfully applied our method to binary class problems, it is also interesting to see how δ modifies the learning behavior. Fig. 3 illustrates the changes of regularization parameter with respect to the training iteration in three binary class problems. In contrast to the multi-class cases shown in Fig. 2, the regularization parameter depicts no consistent increasing trend but becomes rather steady in a range. This is mostly due to the fact that δ exclusively dominates the multi-class regularization term. As $\delta \leq 1$, the classifier weight α is slightly suppressed, which makes our method performs closely to RUSBoost in binary classification problems as shown in Table 3.

To study the effect of regularization term on performance improvement, we incorporated stratified sampling into AdaBoost.M1

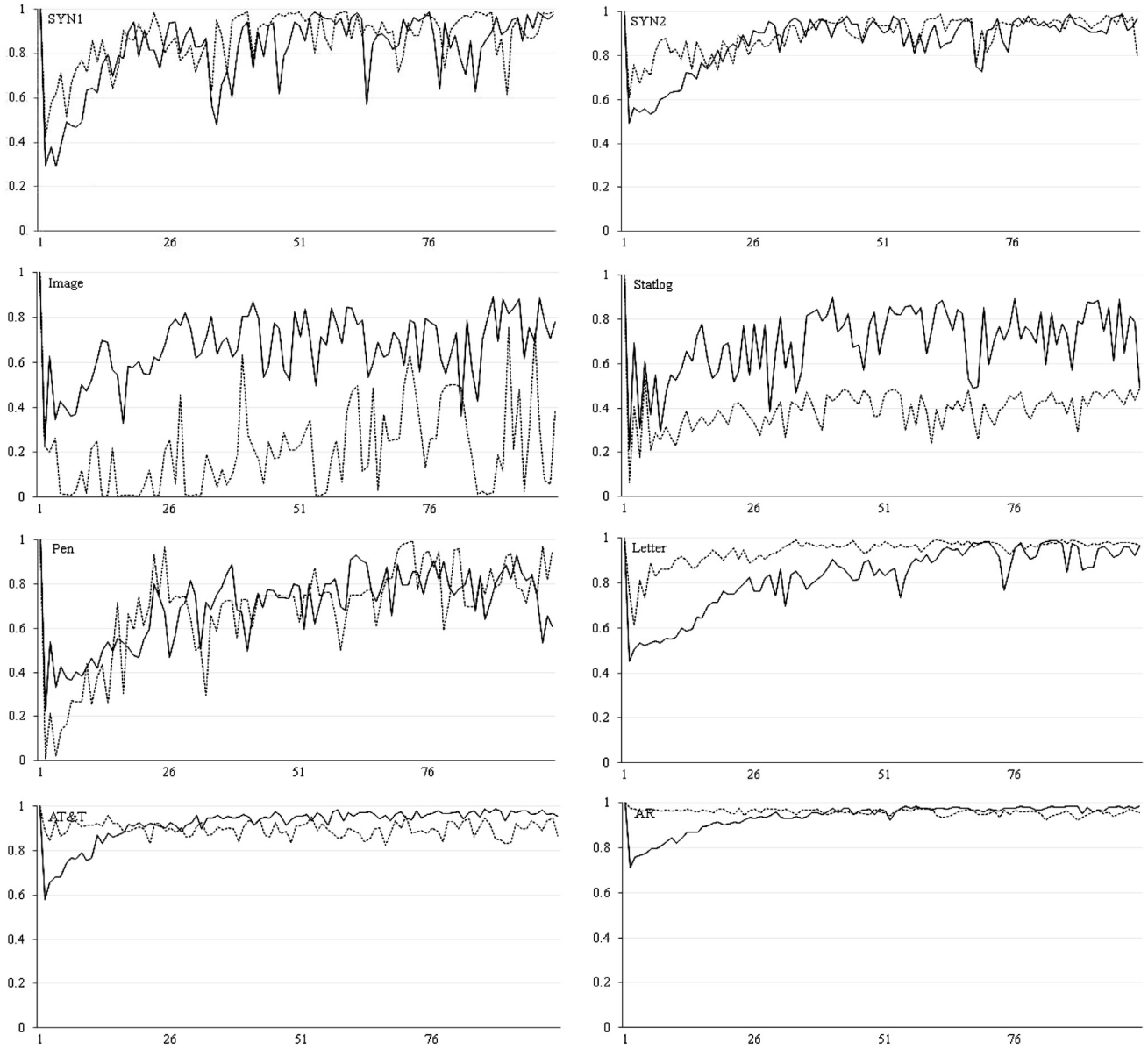

Fig. 2. Regularization parameter δ changes along the training iterations. δ is initialized to one. The solid line depicts the δ of using decision tree as base classifier and the dash line depicts the δ of using neural network as base classifier.

and SAMME. Hence, the difference between these methods and our method mostly results from the regularization term. Table 6 lists the average performance of the three methods with two base learners. The column "imp" gives the improvement of performance in percentage. It is computed by comparing the best results from AdaBoost.M1 and SAMME (with stratified sampling).

As shown in Table 6, the average improvements of our method in sensitivity with the decision tree and neural network as base learners are 5.92% and 3.38%, respectively. More importantly, such improvement is not a result of sacrificing the performance of the majority classes. This is evidential from the improvement in accuracy as shown in Table 6. While the sensitivity of the minority classes improves, the overall accuracy also increases. The average improvements of our method in accuracy with the decision tree and neural network as base learners are 3.94% and 1.91%, respectively. The underlined results are the best performance among all ensemble and learner combinations. Out of 11 test cases, our method results in 9 best sensitivities and 9 best accuracies. Even in the very few cases where our method gives a lower performance, the difference is mostly less than one percent. Clearly, the regularization parameter brings positive impact to the performance of multi-class ensemble.

Cross-referencing with Table 2 we can see that stratified sampling also improves the performance of AdaBoost.M1 and SAMME and in some cases the improvement is great. Note that AdaBoost.M1 produces no results in some cases. Clearly, integrating stratified sampling adds no help to AdaBoost.M1. However, the innate constraint is the stringent error bound that halts the learning process from creating effective classifiers.

Fig. 4 depicts the decision boundaries of three intermediate classifiers using CD data set [1] and the decision tree as base learner. Fig. 4(a) shows the results of AdaBoost.M1 with stratified sampling and Fig. 4(b) shows the results of our method. The minority and majority examples are visualized with circles and asterisks, respec-

[1] The choice of CD data set is simply to minimize the complexity of visualization and focus on the propagation of the learning process.

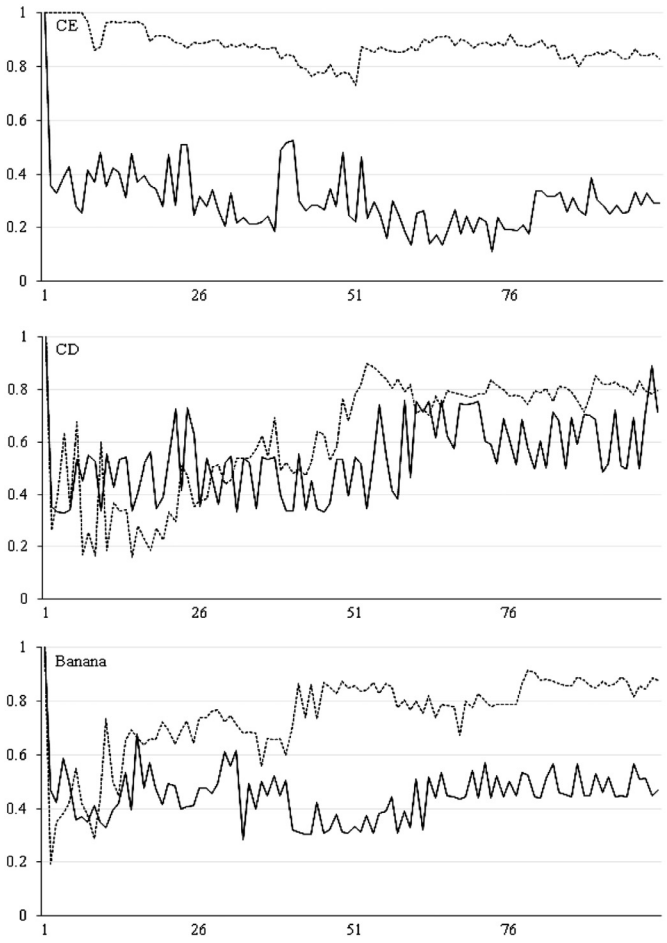

Fig. 3. Regularization parameter δ in binary class problems. The solid line depicts the δ of using decision tree and the dash line depicts the δ of using deep network.

Table 6
Average performance of AdaBoost.M1 and SAMME with stratified sampling and our method. Column "Imp" gives the percentage of improvement using our method. Parenthesis indicates negative number.

Data sets	Decision trees				Deep networks			
Sensitivity	Ada	SAM	REFDL	Imp	Ada	SAM	REFDL	Imp
CE	40.0	39.3	50.0	25	80.0	80.0	82.0	2.5
SYN1	57.1	60.1	58.3	(3)	53.2	63.1	63.6	0.8
SYN2	18.2	42.1	43.1	2.4	–	45.1	45.6	1.1
Image	79.8	82.0	82.9	1.1	88.1	89.0	88.5	(0.6)
Letter	–	57.8	59.5	2.9	60.3	67.0	67.7	1
Pen	84.0	86.5	86.6	0.1	93.2	94.0	94.2	0.2
Statlog	71.5	71.6	73.5	2.7	67.8	64.1	75.3	11.1
AT&T	–	36.2	40.0	10.5	–	58.7	60.0	2.2
AR	–	45.0	46.0	2.2	–	24.0	29.0	20.8
CD	46.7	46.6	54.5	16.7	67.3	67.8	66.3	(2.2)
Banana	51.1	51.6	53.9	4.5	69.2	69.4	69.6	0.3
Accuracy								
CE	69.6	69.8	72.7	4.2	70.3	70.3	73.8	5
SYN1	61.5	57.1	57.3	(0.4)	60.8	61.6	63.0	2.3
SYN2	20.2	34.4	35.7	3.8	–	41.1	40.8	(0.7)
Image	82.3	83.2	83.3	0.1	90.0	90.4	90.8	0.4
Letter	–	54.3	55.9	3	61.7	56.4	57.6	(6.6)
Pen	87.3	87.9	88.1	0.2	96.1	96.1	96.5	0.4
Statlog	77.1	74.8	76.1	(1.3)	78.0	74.9	78.5	0.6
AT&T	–	28.7	35.8	24.7	–	48.1	48.7	1.2
AR	–	27.5	29.0	5.5	–	18.5	22.0	18.9
CD	68.2	68.1	72.0	2.6	82.5	82.3	81.9	(0.7)
Banana	74.2	74.7	75.4	0.9	83.8	83.8	84.0	0.2

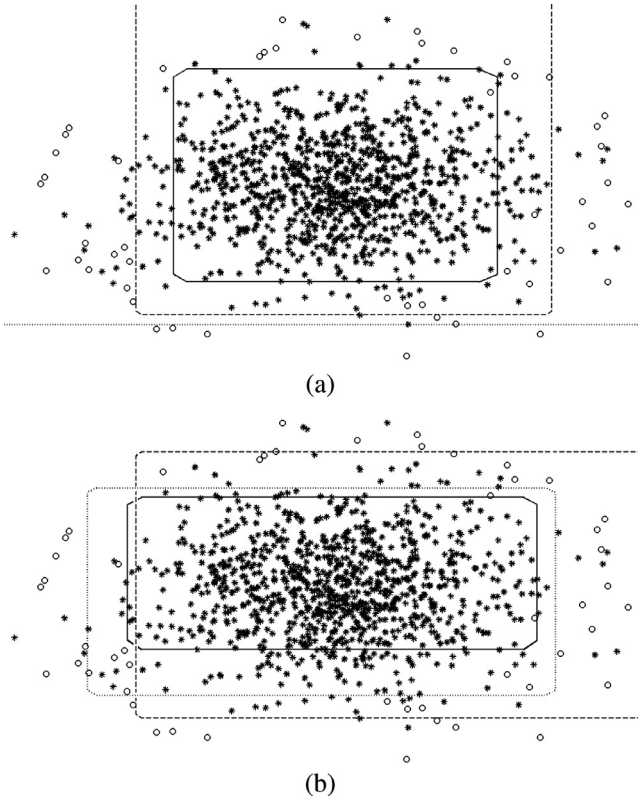

Fig. 4. Decision boundaries of three consecutive classifiers using (a) stratified sampling with AdaBoost.M1 and (b) our method. Solid line draws the decision boundary for the first classifier; dash line draws the decision boundary for the second classifier; and dotted line draws the decision boundary for the third classifier.

tively. Using a subset of examples for training, AdaBoost.M1 starts with a good model that missed only one minority example. However, the dominating majority examples quickly migrate the following classifiers to minimize the count of misclassification by sacrificing the much less number of minority examples.

In contrast, our method demonstrates great robustness using the regularization parameter. As classifiers are developed, each of them attends part of the data distribution and the aggregation of them clearly shows the non-linear margin between classes. The bias of an overwhelmingly large number of majority examples is greatly suppressed.

5. Conclusion

It is common for real-world applications to have an uneven number of examples among multiple classes. The data imbalance, however, usually complicates the learning process, especially for the minority classes, and results in deteriorated performance. In this article, we propose a regularized ensemble framework of deep learning method to handle the imbalanced, multi-class learning problems. This method uses stratified undersampling to recover the balance among classes and addresses the unpredictability of base learners with regularization. The sampling procedure randomly selects examples for the majority classes based on their data distribution and the regularization modifies the loss function to penalize the classifiers with second-round-misclassified examples. This regularization parameter also adjusts the error bound in accordance with classifier's performance.

Experiments are conducted using 11 diverse synthetic and real-world data sets with moderate to high imbalance ratio. Our experiments aim to evaluate the capability and stability of the proposed method. The results demonstrate the superior performance of our

method compared to several state-of-the-art algorithms for imbalanced, multi-class classification problems. It is evident that the proposed method achieved the highest sensitivity for the minority classes in most cases (the best sensitivity in 7 out of 8 multi-class cases). More importantly, our sensitivity gain of the minority classes is accompanied with the improvement of the overall accuracy for all classes. The standard deviation of the results illustrates the consistency of our method. It is worth noting that although our proposed method uses undersampling strategy the ensemble stability is retained. Despite the failure of training by other methods, our method successfully devises ensembles in all cases. In addition to multi-class problems, our experiments with binary-class problems also reveal the applicability of our method and its performance is highly competitive.

Another unique facet of our evaluation is looking at the number of effective classifiers resulted from the training of ensemble. Our experiments show that the proposed method yielded the largest number of effective classifiers in most cases, which indicates the diversity of the base classifiers in the ensemble. In undersampling-based methods, the number of the effective classifier is also an indicator of overfitting, which could be caused by random undersampling that misses important examples when a fairly small number of examples are needed to balance the training data set.

A major concern of applying boosting method to large data set is the computational cost. With a step of forming training set, our method, as well as RUSBoost and SMOTEBoost, requires extra time. However, the time used to select a subset of examples is outweighed by the training time. It turns out that the sampling process allows our method to gain the best efficiency. The reduction in computational cost is significant and in some cases, the improvement is more than 50%. As the volume of training data increase, the gain of efficiency with our method becomes more significant.

The regularization parameter plays a vital role in the algorithm's capability of handling multi-class data sets and improves performance. The regularization penalizes the weights of the base classifiers when they exhibit second-round-misclassified examples especially those with increased weights near the borderline. With different training examples, the regularization term fluctuates in the training process. As training continues, the regularization term enlarges asymptotically towards one. By integrating stratified sampling with AdaBoost.M1 and SAMME, we examined the performance impact from regularization only. The improvements with different base learners differ and the average accuracy enhancement is in the order of 1.91% to 3.94% with maximum improvement at 24.7%.

References

[1] H. He, E.A. Garcia, Learning from imbalanced data, IEEE Trans. Knowl. Data Eng. 21 (9) (2009) 1263–1284.
[2] G.M. Weiss, Mining with rarity: a unifying framework, SIGKDD Explor. 6 (1) (2004) 7–19.
[3] N.V. Chawla, N. Japkowicz, A. Kotcz, Editorial: special issue on learning from imbalanced data sets, SIGKDD Explor. 6 (1) (2004) 1–6.
[4] S. Wang, Z. Li, W. Chao, Q. Cao, Applying adaptive over-sampling technique based on data density and cost-sensitive svm to imbalanced learning, in: The International Joint Conference on Neural Networks, 2012, pp. 1–8.
[5] H. Han, W.-Y. Wang, B.-H. Mao, Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning, in: International Conference on Intelligent Computing, Springer-Verlag, 2005, pp. 878–887.
[6] A. Fernandez, V. Lopez, M. Galar, M.J. del Jesus, F. Herrera, Analysing the classification of imbalanced data-sets with multiple classes: binarization techniques and ad-hoc approaches, Knowl. Based Syst. 42 (2013) 97–110.
[7] Y. Freund, R.E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, J. Comput. Syst. Sci. 55 (119–139) (1997).
[8] R.E. Schapire, Y. Singer, Improved boosting algorithms using confidence-rated predictions, in: Machine Learning, 1999, pp. 80–91.
[9] J. Zhu, H. Zou, S. Rosset, T. Hastie, Multi-class adaboost, Stat. Interface 2 (3) (2009) 349–360.
[10] I. Mukherjee, R.E. Schapire, A theory of multiclass boosting, J. Mach. Learn. Res. 14 (2013) 437–497.
[11] M. Saberian, N. Vasconcelos, Multiclass boosting: theory and algorithms, NIPS, 2011.
[12] N.C.F. Codella, Q.B. Nguyen, S. Pankanti, D.A. Gutman, B. Helba, A.C. Halpern, J.R. Smith, Deep learning ensembles for melanoma recognition in dermoscopy images, IBM J. Res. Dev. 61 (4) (2017) 5:1–5:15.
[13] Y. Xiao, J. Wu, Z. Lin, X. Zhao, A deep learning-based multi-model ensemble method for cancer prediction, Comput. Methods Programs Biomed. 153 (2018) 1–9.
[14] N.V. Chawla, A. Lazarevic, L.O. Hall, K.W. Bowyer, SMOTEBoost: improving prediction of the minority, in: Seventh European Conference on Principles and Practice of Knowledge Discovery in Databases, 2003, pp. 107–119.
[15] C. Seiffert, T.M. Khoshgoftaar, J.V. Hulse, A. Napolitano, RUSBoost: a hybrid approach to alleviating class imbalance, IEEE Trans. Syst. Man Cybern. Part A 40 (1) (2010) 185–197.
[16] O.J. Geiler, L. Hong, G. Yue-jian, An adaptive sampling ensemble classifier for learning from imbalanced data sets, in: International MultiConference of Engineers and Computer Scientists, Hongkong, China, 2010.
[17] S. Wang, X. Yao, Multiclass imbalance problems: analysis and potential solutions, IEEE Trans. Syst. Man Cybern. Part B 42 (4) (2012) 1119–1130.
[18] X. Yuan, M. Abouelenien, A multi-class boosting method for learning from imbalanced data, Int. J. Granul. Comput. Rough Sets Intell. Syst. 4 (1) (2015) 13–29.
[19] M. Kubat, S. Matwin, Addressing the curse of imbalanced data sets: one sided sampling, in: 14th International Conference on Machine Learning, 1997, pp. 179–186.
[20] S. Barua, M. Islam, X. Yao, K. Murase, MWMOTE - majority weighted minority oversampling technique for imbalanced data set learning, IEEE Trans. Knowl. Data Eng. 26 (2) (2012) 405–425.
[21] H. Cao, X.-L. Li, Y.-K. Woon, S.-K. Ng, Integrated oversampling for imbalanced time series classification, IEEE Trans. Knowl. Data Eng. 25 (12) (2013) 2809–2822.
[22] V. Lopez, A. Fernandez, J.G. Moreno-Torres, F. Herrera, Analysis of preprocessing vs. cost-sensitive learning for imbalanced classification. open problems on intrinsic data characteristics, Expert Syst. Appl. 39 (7) (2012) 6585–6608.
[23] S.-J. Yen, Y.-S. Lee, Cluster-based under-sampling approaches for imbalanced data distributions, Expert Syst. Appl. 36 (3) (2009) 5718–5727.
[24] R. Batuwita, V. Palade, Efficient resampling methods for training support vector machines with imbalanced datasets, in: The International Joint Conference on Neural Networks, Barcelona, Spain, 2010, pp. 1–8.
[25] L. Zhou, Performance of corporate bankruptcy prediction models on imbalanced dataset: the effect of sampling methods, Knowl. Based Syst. 41 (2013) 16–25.
[26] L. Piras, G. Giacinto, Synthetic pattern generation for imbalanced learning in image retrieval, Pattern Recognit. Lett. 33 (16) (2012) 2198–2205.
[27] M. Wasikowski, X. wen Chen, Combating the small sample class imbalance problem using feature selection, IEEE Trans. Knowl. Data Eng. 22 (10) (2010) 1388–1400.
[28] D. Drown, T. Khoshgoftaar, N. Seliya, Evolutionary sampling and software quality modeling of high-assurance systems, IEEE Trans. Syst. Man Cybern. Part A 39 (5) (2009) 1097–1107.
[29] S. Garcia, J. Derrac, I. Triguero, C. Carmona, F. Herrera, Evolutionary-based selection of generalized instances for imbalanced classification, Knowl. Based Syst. 25 (1) (2012) 3–12.
[30] H. Yu, J. Ni, J. Zhao, ACOSampling: an ant colony optimization-based undersampling method for classifying imbalanced DNA microarray data, Neurocomputing 101 (2013) 309–318.
[31] G. Karakoulas, J. Shawe-Taylor, Optimizing classifiers for imbalanced training sets, in: The Conference on Advances in Neural Information Processing Systems II, MIT Press, Cambridge, MA, USA, 1999, pp. 253–259.
[32] S. Chen, H. He, E. Garcia, RAMOBoost: ranked minority oversampling in boosting, IEEE Trans. Neural Netw. 21 (10) (2010) 1624–1642.
[33] Y. Liu, X. Yu, J.X. Huang, A. An, Combining integrated sampling with svm ensembles for learning from imbalanced datasets, Inf. Process. Manage. 47 (4) (2011) 617–631.
[34] H.E. Saadi, A.F. Al-sadek, M.W. Fakhr, Informed under-sampling for enhancing patient specific epileptic seizure detection, Int. J. Comput. Appl. 57 (16) (2012) 41–46.
[35] H. He, Y. Bai, E. Garcia, S. Li, ADASYN: adaptive synthetic sampling approach for imbalanced learning, in: IEEE International Joint Conference on Neural Networks, Hongkong, China, 2008, pp. 1322–1328.
[36] B. Yuan, X. Ma, Sampling + reweighting: boosting the performance of adaboost on imbalanced datasets, in: The International Joint Conference on Neural Networks, Brisbane, QLD, Australia, 2012, pp. 1–6.
[37] Y.L. Murphey, H. Wang, G. Ou, L.A. Feldkamp, OAHO: an effective algorithm for multi-class learning from imbalanced data, in: International Joint Conference on Neural Networks, 2007, pp. 406–411.
[38] A.S. Ghanem, S. Venkatesh, G. West, Multi-class pattern classification in imbalanced data, in: 20th International Conference on Pattern Recognition, 2010, pp. 2881–2884.
[39] A. Ghanem, S. Venkatesh, G. West, Learning in imbalanced relational data, in: the 19th International Conference on Pattern Recognition, 2008, pp. 1–4.
[40] P. Jeatrakul, K.-W. Wong, Enhancing classification performance of multi-class imbalanced data using the OAA-DB algorithm, in: The International Joint Conference on Neural Networks, Brisbane, QLD, Australia, 2012, pp. 1–8.

[41] Z. Qi, B. Wang, Y. Tian, P. Zhang, When ensemble learning meets deep learning: a new deep support vector machine for classification, Knowl. Based Syst. 107 (2016) 54–60.
[42] R. Rasti, M. Teshnehlab, S.L. Phung, Breast cancer diagnosis in dce-mri using mixture ensemble of convolutional neural networks, Pattern Recognit. 72 (2017) 381–390.
[43] M.H. Bae, T. Wu, R. Pan, Mix-ratio sampling: classifying multiclass imbalanced mouse brain images using support vector machine, Expert Syst. Appl. 37 (7) (2010) 4955–4965.
[44] F. Fernández-Navarro, C. Hervás-Martínez, P. Antonio Gutiérrez, A dynamic over-sampling procedure based on sensitivity for multi-class problems, Pattern Recognit. 44 (8) (2011) 1821–1833.
[45] L.-I. Tong, Y.-C. Chang, S.-H. Lin, Determining the optimal re-sampling strategy for a classification model with imbalanced data using design of experiments and response surface methodologies, Expert Syst. Appl. 38 (4) (2011) 4222–4227.
[46] B. Debowski, S. Areibi, G. Grewal, J. Tempelman, A dynamic sampling framework for multi-class imbalanced data, in: The International Conference on Machine Learning and Applications, Boca Raton, FL, USA, 2012, pp. 113–118.
[47] X. Yuan, M. Abouelenien, A boosting method for learning from uneven data for improved face recognition, in: The International Conference on Machine Learning and Applications, Boca Raton, FL, USA, 2012, pp. 119–122.
[48] S. Wang, H. Chen, X. Yao, Negative correlation learning for classification ensembles, in: The International Joint Conference on Neural Networks, Barcelona, Spain, 2010, pp. 1–8.
[49] M. Lichman, UCI Machine Learning Repository, University of California, Irvine, School of Information and Computer Sciences, 2013. http://archive.ics.uci.edu/ml

Dr. Xiaohui Yuan received a B.S. degree in electrical engineering from the Hefei University of Technology, Hefei, China in 1996 and a Ph.D. degree in computer science from Tulane University, New Orleans, LA, USA in 2004. He is currently an Associate Professor at the Department of Computer Science and Engineering in the University of North Texas. His research interests include computer vision, data mining, machine learning, and artificial intelligence. His research findings have been reported in over one hundred peer-reviewed papers. Dr. Yuan is a recipient of Ralph E. Powe Junior Faculty Enhancement award in 2008 and the Air Force Summer Faculty Fellowship in 2011, 2012, and 2013.

Dr. Lijun Xie is received M.S. degree in medicine from Zhejiang University College of Medicine. He is with the Second Affiliated Hospital, Zhejiang University College of Medicine. His specialties include bone tumor treatment, spinal surgery, limb trauma, joint surgery, and rheumatism.

Dr. Mohamed Abouelenien is currently a postdoc research fellow in electrical engineering and computer science department at University of Michigan, Ann Arbor. He received his B.Sc. and M.Sc. degrees in electronics and communication engineering in 2005 and 2008 from the Arab Academy for Science and Technology in Egypt and his Ph.D. from department of computer science and engineering at University of North Texas in 2013. His research interest includes multimodal analysis, deception detection, machine learning, ensemble classification, image processing, pattern recognition, face recognition, computer vision, and dimensionality reduction.