# Measure optimized cost-sensitive neural network ensemble for multiclass imbalance data learning

Peng Cao, Dazhe Zhao
Key Laboratory of Medical Image Computing of Ministry of Education, Northeastern University
Shenyang, China
e-mail: cao.p@neusoft.com, zhaodz@neusoft.com

Osmar Zaiane
Computing Science, University of Alberta
Edmonton, Canada
zaiane@cs.ualberta.ca

***Abstract*—The performance of traditional classification algorithms can be limited on imbalanced datasets. In recent years, the imbalanced data learning problem has drawn significant interest. In this work, we focus on designing modifications to neural network, in order to appropriately tackle the problem of multiclass imbalance. We propose a hybrid method that combines two ideas: diverse random subspace ensemble learning with evolutionary search, to improve the performance of neural network on multiclass imbalanced data. An evolutionary search technique is utilized to optimize the misclassification cost under the guidance of imbalanced data measures. Moreover, the diverse random subspace ensemble employs the minimum overlapping mechanism to provide diversity so as to improve the performance of the learning and optimization of neural network. We have demonstrated experimentally using UCI datasets that our approach can achieve better result than state-of-the-art methods for imbalanced data.**

***Keywords-imbalanced data; cost sensitive learning; ensemble classifier; swarm intelligence***

## I. INTRODUCTION

Class imbalance learning has been recognized as a crucial problem in machine learning and data mining [1-2]. The issue occurs when the training data is not evenly distributed among classes. This problem is particularly critical in many real applications, such as fraud detection and medical diagnoses. In these cases, standard classifiers generally perform poorly. Classifiers usually tend to be overwhelmed by the majority class and ignore the minority class examples. Therefore, we need to improve traditional algorithms so as to handle imbalanced data.

Much work has been done in addressing the class imbalance problem. These methods can be grouped in two categories: the data perspective [3-5] and the algorithm perspective [6-8]. Most existing imbalance data learning approaches are still limited to the binary class imbalance problems. The common approaches have been shown to be less effective or even degrade the outcome when dealing with multiclass tasks [7]. The multi-class imbalance problem poses a new challenge for classification task [7, 9]. It is desirable to develop an effective method specifically to handle the multiclass imbalance issue.

We make the traditional neural network classification algorithm into a cost-sensitive adaptation (CSNN) by injecting the costs of misclassification into the output of posterior probability to overcome the multiclass imbalanced data classification issue. In the construction of cost sensitive learning, the parameter of misclassification cost plays an indispensable role. However, the appropriate cost cannot be made required as input, and the empirical methods are not workable as the search space is expanded exponentially for the multiple classes as the number of imbalanced classes increases. We propose a new algorithm, called EDS (**E**volutionary search combined with **D**iverse random **S**ubspace ensemble), to help CSNN improve the classification performance on multiclass imbalanced data. In EDS, an evolutionary search is used as the searching strategy to effectively search the optimum misclassification costs in the multiclass imbalance scenario according to the objective function. In addition, since combining multiple neural networks can achieve stronger generalization ability [10], we extend the random subspace method [11] to enhance the diversity by employing the minimum overlapping mechanism, so as to avoid overfitting in the procedure of the learning and optimization of CSNN. Furthermore, the ensemble can determine the optimal amount of non-redundant components automatically.

## II. COST SENSITIVE NEURAL NETWORK

The typical way of handling imbalanced training data is by re-sampling. The main disadvantage of re-sampling techniques is that they may cause loss of important information or model overfitting, as they change the data distribution. Assigning distinct costs to the training examples seems to be the most effective approach to deal with the class imbalance problem. The cost-sensitive learning takes misclassification costs into account during the model construction, and does not modify the imbalanced data distribution directly. The standard neural network is cost insensitive. In standard neural network classifiers, the class returned is $C^*$ by comparing the probability of each class directly for each instance $x$ according to Eq.(1).

$$C^* = \underset{C\in\{1,\ldots,M\}}{argmax}(p_1(C_1 \mid x),\ldots, p_M(C_M \mid x)) \quad (1)$$

where $P_i$ denotes the probability value of each class from the neural network, $\sum_{i=1}^{M} P_i = 1$ and $0 \le P_i \le 1$. $M$ is the number of the class.

The probabilities generated by the standard neural network are biased in the imbalanced data distribution. To improve the recognition of the minority class, the probability that a sample belongs to a certain class is replaced with the altered probability, which takes the misclassification costs into account, is found to be relatively a good choice in training CSNN [7]. This method uses the training set to construct a neural network, and the cost sensitivity strategy is introduced in the test phase. Given a certain cost matrix, the CSNN returns the class $C^*$, which is computed by injecting the cost according to Eq.(2).

$$\begin{aligned} C^* &= \underset{C\in\{1,\ldots,M\}}{argmax}(p_1^*(C_1 \mid x),\ldots,p_M^*(C_M \mid x)) \\ &= \underset{C\in\{1,\ldots,M\}}{argmax}(\eta_1 cost(C_1)p(C_1 \mid x),\ldots,\eta_M cost(C_M)p(C_M \mid x)) \end{aligned} \tag{2}$$

where $\eta_i$ is a normalization term such that $\sum_{i=1}^{M} P_i^* = 1$ and $0 \le P_i^* \le 1$.

## III. THE MEASURE OPTIMIZED CSNN

### A. Evolutionary search CSNN (ECSNN)

For binary classes ($M$=2), we can iteratively search the best cost parameter for which the evaluation measure is maximized [6]. However for a multiclass application ($M$>2), it is difficult to select the appropriate cost vector in the expanded space. Hence, searching an efficient cost setup becomes a critical issue for applying the cost sensitive neural network to multiclass applications. We propose a method called ECSNN (**E**volutionary search CSNN), which employs an evolutionary technique to carry out the meta-learning for searching an optimal class cost setup, which will be applied to the CSNN algorithm improving the classification performance of the imbalanced datasets.

The popularity of evolutionary search has also instigated the development of numerous data mining algorithms [12]. In an evolutionary search approach, individuals of a swarm move through a solution space and look for solutions for the data mining task at hand. We utilize the evolutionary search method as the optimization strategy to search the cost. The Artificial Bee Colony (ABC) algorithm is a swarm-based algorithm that was introduced based on the intelligent foraging behavior of honey bee swarms [13]. In the ABC algorithm, the position of a food source represents a possible solution to the optimization problem and the nectar amount of the food source corresponds to the quality (fitness) of the associated solution. It consists in a set of possible solutions $x_i$ that is represented by the position of the food sources. On the other hand, in order to find the best solution, three classes of bees are used: employed bees, onlooker bees and scout bees. These bees have different tasks in the colony.

In order to produce a candidate food position $v_i$ in the neighborhood of its current source $x_i$, the following expression is used and compares the new one against the current solution.

$$v_{ij} = x_{ij} + \varphi_{ij}(x_{ij} - x_{kj}) \tag{3}$$

where $k \in \{1\ldots,SN\}$ and $j \in \{1\ldots,D\}$ are randomly chosen index. $SN$ denotes the size of the solution; $j$ is the dimension of each solution. $\varphi_{ij}$ is a random number between [-1, 1].

An artificial onlooker bee chooses a food source depending on the probability value associated with that food source, calculated by the following expression:

$$p_i = fit_i \Big/ \sum_{n=1}^{SN} fit_n \tag{4}$$

where $fit_i$ is the fitness value of the *i-th* solution. The value of predetermined number of cycles is an important control parameter of the ABC algorithm, which is called ''*limit*" for abandonment. Assuming that the abandoned source is $x_i$ then the scout discovers a new food source to be replaced with $x$. This operation can be defined as:

$$x_i^j = x_{\min}^j + rand[0,1] \times (x_{\max}^j - x_{\min}^j) \tag{5}$$

Where $x_{min}^j$ and $x_{max}^j$ are lower and upper bounds of parameter $j$.

The process is repeated through a predetermined number of cycles, called Maximum Number of Cycle (*MCN*). From the results obtained in [14], it can be concluded that the performance of the ABC algorithm is better than or similar to that of a Genetic algorithm and Particle swarm optimization.

The ABC algorithm can optimize the proper class cost parameters based on the posterior probabilities produced by the neural network to maximize the performance measure. According to the specific objective function of artificial bee colony, we can optimize the class cost vector to achieve the optimum classification performance. In the procedure of searching for the best class cost vector, evaluation measures play a crucial role in both assessing the classification performance and guiding the modeling of the classifier. For imbalanced datasets, the evaluation metric should take into account the imbalance. The average accuracy is not an appropriate evaluation metric. We used G-mean as the fitness of the ABC algorithm to guide the optimization process. The G-mean is the geometric mean of accuracies measured separately on each class, which is commonly utilized when performance of both classes is concerned and expected to be high simultaneously.

$$G-mean = (\prod_{i}^{M} Acc_i)^{1/M} \tag{6}$$

We set the cost values for instances in the same class with an identical value. Let $Cost(C_i)$ denote the cost value of class $i$. Cost values of $M$ classes then make up a cost vector [$Cost(C_1)$, $Cost(C_2)$, …, $Cost(C_M)$]. Since only the ratio of costs is effective, we set the cost of the largest class to 1, thus there are $M$-1degrees of freedom. This cost vector is encoded in each bee.

In order to optimize class cost parameters while avoiding overfitting, the training data is partitioned into two subsets, training subset and validation subset. The training subset is

used to construct the cost sensitive neural network classifier that yields posterior probabilities, while the validation subset is used for obtain the optimal cost vector. The detailed algorithm of ECSNN is shown in **Algorithm 1.** The wrapper paradigm can discover the *bestCostVector* for a dataset based on optimizing evaluation functions. When predicting test instance, all the class posterior probabilities have to be multiplied with the corresponding cost parameter. The algorithm belongs to the post-processing strategy, which does not change the data distribution and distort the information like re-sampling techniques.

**Algorithm** 1 *ECSNN*

**Input**: Training set *TrainingSet*, Test set *TestSet,* Maximum Cycle Number *MCN*; Population size *SN*; Limit *limit*

**Training phase:**

1. Separate *TrainingSet* into *TrSet* (80%) for training and *ValSet* (20%) for validation
2. Construct a classifier *L* on the *TrSet;*
3. Predict probability estimates on the *ValSet*
4. Initialize the food source positions $x_i$, $i$ =1...*SN*
5. Obtain the G-mean $GM_i$ with the *CostVector* optimized in $x_i$
6. *cycle=1*

**repeat**

7. Produce new solutions $x_i'$ for the employed bee and evaluate them with G-mean
8. Apply the greedy selection process
9. Calculate the probability $p_i$ for the solutions $x_i'$ with Eq.4
10. Produce new solutions $v_i$ with Eq.3 for the onlooker from solutions $x_i'$ selected depending on $p_i$ and evaluate them
11. Apply the greedy selection process
12. Determine the abandoned solution for the scout according to the value of *limit*, if exists, and replace it with a new randomly produced solution with Eq.5
13. Memorize the best solution achieved so far; *cycle*++

**until** cycle=*MCN*

14. Obtain the *bestCostVector* from the best solution

**Test phase:**

15. Generate probabilities of each instance on the *TestSet* with *L*
16. Obtain the class of each instance with *bestCostVector* using **Eq**.(2)

### *B. ECSNN combined with diverse random subspace ensemble (EDS-CSNN)*

The ECSNN algorithm is based on the outputs of neural network on the whole data. The performance of ECSNN depends on the outputs of posterior probabilities generated by the neural network. However, the outputs generated may be inaccurate due to irrelevant features or noisy instances. Moreover, the imbalanced problem is often accompanied by high dimensional data in the practical domain [15]. All the issues can decrease the performance of ECSNN. Therefore, we propose an improvement of random subspace ensemble, diverse random subspace that can provide an effective framework to improve the performance of ECSNN.

The use of different spaces for ensemble construction has been extensively explained in recent research. Ho showed that the random subspace was able to improve the generalization error [11]. In the random subspace method, individual classifiers are built by randomly projecting the original data into feature subspaces and training a proper base-learner on these subspaces. Since the random subspace combines multiple classifiers of this type, each with a random bias based on the features it sees, random subspace often prove more effective than learning the base classifier on all of the features.

Since diversity is known to be an important factor affecting the generalization performance of ensemble methods, several means have been proposed to get varied base-classifiers inside an ensemble. We propose an improvement of random subspace method, called diverse random subspace in order to obtain more diversity in each classifier. The common random subspace method is extended by integrating bootstrapping samples to obtain more diversity in each classifier at first. In the bootstrapping method, different training subsets are generated with uniform random selection with replacement. In addition, in the random subspace method, different features in each training subsets are randomly chosen for producing component classifiers. However, this cannot ensure the diversity of each subset since the instances and the features are chosen randomly. Therefore, for improving diversity between each subset, we use a formulation to make sure each subset is diverse. Firstly, we introduce a concept of *overlapping rate:*

$$Overlapping\ rate = \frac{subSet_i \cap subSet_j}{N_{fea} \times N_{ins}} \tag{7}$$

where the *subset* is the subset within a certain subspace, $N_{fea}$ and $N_{ins}$ are the feature size and instance size of each *subset.* All the subsets are of equal size.

In addition, we guarantee that the class ratio of each subset follows the one of the original training data distribution. We quantify data diversity between each subset with the data overlapping region, which measures the proportion of feature and instance subspace overlap between the training data of different classifiers in the ensemble. We then introduce a threshold $T_{over}$ to control the intersection between each subset. The overlapping rate of all the subsets needs to be smaller than the threshold $T_{over}$. Therefore $T_{over}$ is critical to the performance of the ensemble. If it is too large, the subsets lack diversity. If it is too low, the ensemble size is small, diminishing the advantage of ensemble classification. It is a trade-off between the diversity and the required ensemble size.

Through quantifying data diversity between each subset for a component classifier with the data overlapping region which measures the proportion of feature and instance subspace overlap between the training data of different classifiers in the ensemble, we can guarantee the diversity of subsets provided to each classifier, and at the same time provide a way to adaptively determine in an iterative way the number of classifiers in the ensemble. The

*GenerateDiverseSets* algorithm can be described as in **Algorithm 2**. The function ***isDiverse***($D_s^k$, *DiverseSet*, $T_{over}$) examines if the new projection $D_s^k$ is diverse enough from the previously collected projections in *DiverseSet* based on the overlapping region threshold $T_{over}$. The generation of projections stops when there is stagnation – i.e. after enough trials, no new projection is diverse enough from the collected subspaces. Hence, the number of individual classifiers is determined dynamically.

**Algorithm** 2 *GenerateDiverseSets*

**Input**: Training set *TrainingSet*, Ratio of bootstrap samples $R_s$, Ratio of feature subspace $R_f$, Overlapping region threshold $T_{over}$, Stagnation rate *sr=100*

1. *change*=0; *DiverseSets*={};
   ***while*** *change*<*sr* **do**
2. A bootstrap sample $D_s$ selected with replacement from *TrainingSet* with $R_s$
3. Generate subset $D_s^k$ by selecting a random subspace with $R_f$
   ***if isDiverse***($D_s^k$, *DiverseSets*, $T_{over}$)==true
4. ***then*** *DiverseSets->add*($D_s^k$); *change*=0;
5. ***else*** *change*=*change*+1;
   ***end while***

**Output**: *DiverseSets*

**Algorithm** 3 *EDS-CSNN*

**Input**: Training set *TrainingSet*, Test set *TestSet,* Ratio of bootstrap samples $R_s$, Ratio of feature subspace $R_f$, Overlapping region threshold $T_{over}$,

**Training phase:**

1. *DiverseSets* = ***GenerateDiverseSets***(*TrainingSet*, $R_s$, $R_f$, $T_{over}$);
   ***for*** each subset $D_k$ in *DiverseSets*
2. Construct a classifier $L_k$ in $D_k$
3. Obtain *bestCostVector* according to Algorithm 1.
4. $L_k$->*Subspace*= *subspace*($D_k$)
5. $L_k$->*bestCostVector* = *bestCostVector*
6. *Ensemble*=*Ensemble* $\cup$ $L_k$
   ***end for***

**Testing phase:**

7. Calculate output from each classifier $L_k$ of *Ensemble* with its *bestCostVector* in its *Subspace* on the *TestSet*
8. Generate the final output by aggregating all the outputs

After obtaining the diverse set, both the construction of the neural network and optimization of class costs are conducted on each subset instead on the whole data, then ultimately combine classifiers with different characteristics and achieve improved performance. Therefore, varying the feature subsets gives an opportunity to control the diversity of the feature sets provided to each classifier in the ensemble and capture possible patterns that are informative on classification. The procedure of training and optimization of CSNN can be carried out in parallel to reduce the learning time. In addition, each CSNN classifier is modeled in the reduced subset with fewer instances and features. Hence the computational time is acceptable. **Algorithm 3** illustrates the EDS-CSNN algorithm.

## IV. EXPERIMENTS

### A. Experiment design

We choose 9 publicly available datasets from different domains with varying levels of class imbalance and number of classes. **Table 1** shows the varying characteristics of the datasets. In our empirical experiments, the Back-propagation neural network is employed as the base learner. The number of input neurons is equal to the number of features for each dataset, and the number of neurons in the hidden layer is set to be 10. The inner training epochs is set to be 200 with a learning rate of 0.1. All of methods in the comparison were implemented in Java on the platform of the WEKA. Like most of the optimization algorithms, the ABC algorithm also has control parameters to be set before running the algorithm. We set the *SN* twice as many as the number of classes in our experiment. Through experiments, we found that the optimization can get a good and stable solution before the 500 cycles, thus the *MCN* is set to 500. We set the *limit* as the square of the number of class, which is generally appropriate.

**Table 1.** The data sets used for experimentation

| Dataset | *C* | *Inst.* | *F* | *Class distribution* |
|---|---|---|---|---|
| Cmc | 3 | 1473 | 9 | 629/333/511 |
| Balance | 3 | 625 | 4 | 49/288/288 |
| Car | 4 | 1728 | 6 | 1210/384/69/65 |
| Annealing | 4 | 898 | 38 | 8/99/684/67/40 |
| Page | 5 | 5473 | 10 | 4913/329/28/88/115 |
| Cleveland | 5 | 303 | 13 | 164/55/36/35/13 |
| Glass | 6 | 214 | 9 | 70/76/17/13/9/29 |
| Satimage | 6 | 6435 | 36 | 1533/703/1358/626/707/1508 |
| Yeast | 10 | 1484 | 9 | 463/429/244/163/51/44/35/30/20/5 |

### B. Evaluating the effectiveness of EDS-CSNN

We assess the performance of the EDS-CSNN algorithm and compare it with the original random subspace (ES-CSNN) as well as the single ECSNN. In ES-CSNN and ECSNN, each training set is separated randomly into training subset (70%) for training CSNN and validation subset (30%) for optimizing cost parameters. The ensemble size of ES-CSNN is set to 50. In the construction of EDS-CSNN, we enforce the independence of each subset by minimizing the overlapping region among the subsets for each classifier in the ensemble. This approach allows us to determine the ensemble size adaptively with a certain overlapping region threshold. Since ES-CSNN has a limit on the ensemble size, to have a fair comparison, we set the maximum of the ensemble size of diverse random subspace to the same limit, typically fixed at 50. To that end, we selected the first 50 from the *DiverSets* if the limit is exceeded. In diverse random subspace, the ratio parameters are under the default condition where the ratio of bootstrap sampling $R_s$ is 0.7 and the ratio of features $R_f$ is 0.5. Here we vary the value of $T_{over}$ to exploit the relationship between $T_{over}$ and classification performance.

The range of $T_{over}$ is [0.2, 0.5], the step is 0.02. For each value of $T_{over}$, 10-fold cross validation is conducted to obtain the average value. For space considerations, we only show the results on Page dataset in **Figure 2**.

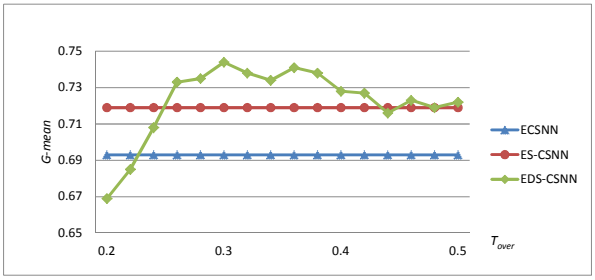


**Figure 2.** The G-mean when varying the threshold parameter $T_{over}$

**Figure 2** shows the performance of single ECSNN, ES-CSNN and EDS-CSNN with varying $T_{over}$. We can see the result of G-mean changes as we vary the value of $T_{over}$ in EDS-CSNN. A smaller $T_{over}$ represents stronger diversity in the subsets. Increasing the value of $T_{over}$ loosens the restriction of diversity. EDS-CSNN can outperform the single ECSNN and ES-CSNN when $T_{over}$ gets to a certain value. We can see that the best $T_{over}$ was 30% for Page dataset and the corresponding G-mean value is 0.744.

Clearly, this value of $T_{over}$ should not be constant as the optimal value of $T_{over}$ depends on the distribution of the dataset. It determines the diversity and number of components, so as to affect the final performance directly. Therefore, we have to estimate the optimal parameter for obtaining the best performance on each dataset. In order to estimate the optimal $T_{over}$, it is chosen by cross validation in the data set. We randomly divided the training data set into two sets: the training subset (80%) and the validation subset (20%) for measuring the performance of each $T_{over}$. This process is repeated 10 times, and then an average G-mean result for each $T_{over}$ is obtained. The output is a $T_{over}$ which obtains the best G-mean value among all tests. We also optimize the optimal $T_{over}$ for diverse random subspace framework methods combined with traditional neural network (NN) and CSNN with default cost setup separately. In the default CSNN, the misclassification cost for class $C_i$ is set to $ImbaRatio_i$ which is the size ratio between the largest class and each class $C_i$. After obtaining the individual optimal $T_{over}$ for diverse random subspace framework; we compare the performance of the three different types of neural network classifiers working on the single model, original random subspace with a fixed ensemble size, as well as our diverse random subspace separately. All the experiments are carried out by 10-fold cross-validation. For the diverse random subspace framework, the section of the 10 fold cross validation is totally independent from the one of cross validation for obtaining the optimal $T_{over}$.

It is clear from the experimental results in **Table 2** that EDS-CSNN obtained the best result on the majority datasets. This indicates that diverse random subspace ensemble with evolutionary search optimization is a very effective strategy for cost sensitive neural network. Diverse random subspace emphasizes ensemble diversity during training, so as to enhance the learning of neural network and optimization of cost parameters. Moreover, each CSNN constructed in the individual different subset can find potentially interesting local data characteristic. Especially for the datasets with high dimensional feature such as *Annealing* and *Satimage*, EDS-CSNN offers a great advantage over other solutions. Furthermore, the diverse subsets construction can achieve a better performance than the complete ensemble on the imbalanced data. The result indicates that the diversity in the ensemble can facilitate class imbalance learning. However, diverse random subspace ensemble cannot achieve the best result on the low dimensional dataset, such as Balance which has 4 attributes. That is because the random subspace method is more effective when datasets have a large numbers attributes. With a very small number of attributes, each classifier receives a small set of features and thus is weak.

Furthermore, regardless of the single or ensemble model, the ABC optimization improves the performance of CSNN. It demonstrates the setting of cost based on the prior class distributions is not appropriate. The optimization method can achieve the optimum cost under the guidance of G-mean, so as to achieve the best performance. **Table 2** also lists the optimal $T_{over}$ value and the corresponding ensemble size of EDS-CSNN. We found that the size is smaller than 50 on the majority cases. The results reveal that diverse random subspace can generate an ensemble model with smaller sizes but stronger generalization ability [10].

**Table 2.** The comparative results of the neural network methods in terms of G-mean on the multiclass class

| **Datasets** | **Single** | | | **Random subspace** | | | **Diverse random subspace** | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | **NN** | **CSNN** | **ECSNN** | **NN** | **CSNN** | **ECSNN** | **NN** | **CSNN** | **ECSNN** | | |
| | **G-mean** | | | **G-mean** | | | **G-mean** | | **G-mean** | $T_{over}$ | **size** |
| Cmc | 0.714 | 0.745 | 0.723 | 0.738 | 0.763 | 0.744 | 0.738 | 0.764 | **0.783** | 0.44 | 36 |
| Balance | 0 | 0.214 | **0.557** | 0 | 0.232 | 0.528 | 0 | 0.226 | 0.532 | 0.32 | 29 |
| Car | 0.752 | 0.815 | 0.837 | 0.766 | 0.827 | 0.853 | 0.807 | 0.834 | **0.879** | 0.36 | 35 |
| Annealing | 0.921 | 0.930 | 0.928 | 0.939 | 0.945 | 0.948 | 0.933 | 0.947 | **0.961** | 0.5 | 50 |
| Page | 0.654 | 0.693 | 0.703 | 0.659 | 0.701 | 0.722 | 0.688 | 0.718 | **0.741** | 0.3 | 31 |
| Cleveland | 0.171 | 0.186 | 0.221 | 0.164 | 0.191 | 0.235 | 0.168 | 0.197 | **0.264** | 0.28 | 43 |
| Glass | 0.374 | 0.424 | 0.535 | 0.429 | 0.438 | **0.589** | 0.426 | 0.440 | 0.578 | 0.28 | 28 |
| Satimage | 0.786 | 0.806 | 0.821 | 0.827 | 0.842 | 0.878 | 0.833 | 0.842 | **0.894** | 0.5 | 50 |
| Yeast | 0 | 0.232 | 0.278 | 0 | 0.323 | 0.385 | 0.254 | 0 | **0.406** | 0.38 | 36 |

### C. Comparing EDS-CSNN with the-state-of-art methods

After finding out that EDS-CSNN can improve the classification performance of neural network, we empirically assessed the algorithm against the state-of-the art methods for multiclass imbalanced data learning, such as Editing Nearest Neighbour rule under-sampling (ENN) [5], SMOTEBoost (SMB) [3], Random subspace method combined with SMOTE (SM-RSM) [4] and MetaCost (MC) [8]. ENN does not require a user specified under-sampling ratio and $K$ is set to 3. For SMB and SM-RSM, the amount of new data for a class $C_i$ is set to be the size difference between the largest class and class $C_i$. In the setting of MC, the misclassification cost for class $C_i$ is set to the size ratio between the largest class and each class $C_i$. The sizes of components are 50 in the all ensemble classifiers. **Table 3** summarizes the performance of the compared algorithms, in which the best performance for each dataset is highlighted.

**Table** 3. G-mean on 9 UCI datasets for several classification methods

| Datasets | MC | ENN | SMB | SM-RSM | EDS-CSNN |
|---|---|---|---|---|---|
| Cmc | 0.685 | 0.719 | 0.749 | 0.742 | **0.783** |
| Balance | 0.348 | 0 | **0.542** | 0.514 | 0.532 |
| Annealing | 0.928 | 0.928 | 0.931 | 0.954 | **0.961** |
| Nursery | 0.759 | 0.758 | **0.811** | 0.802 | 0.798 |
| Page | 0.737 | 0.684 | 0.739 | **0.741** | **0.741** |
| Cleveland | 0.204 | 0.027 | 0.142 | 0.075 | **0.264** |
| Glass | 0.515 | 0.425 | 0.557 | 0.561 | **0.578** |
| Satimage | 0.834 | 0.825 | 0.841 | 0.864 | **0.894** |
| Yeast | 0.138 | 0 | 0.327 | 0.263 | **0.406** |

ENN is the worst method since it is hard to identify the noise when the distribution is complex and imbalanced. Some border points may also be removed as noise while they are useful for training, resulting in loss of information. Both SMB and SM-RSM benefit from the diversity of the ensemble framework. However they manipulate the instances blindly without taking the majority class into consideration, resulting in generating noise instance. MetaCost performs slightly worse than the other advanced methods. It may be because the ratio misclassification cost based on the size ratio between two classes is not appropriate, which reveals again that the misclassification cost is vital for cost sensitive learning, and needs to be searched by some heuristic methods.

## V. CONCLUSIONS

In practice, many problem domains have more than two classes with uneven distributions, but there are fewer solutions in multiclass imbalance problems. In this paper, we present a hybrid system that combines neural networks, random sub-space ensembles and evolutionary search into EDS, a framework for cost sensitive neural network in order to advance the classification of multiclass imbalanced data. The key characteristics of EDS are cost searching and ensemble learning. In EDS, ABC algorithm is employed to search the optimal misclassification cost parameters from the available data; and diverse random subspace offers a good framework for imbalanced data learning as it produces varied and complementary base classifiers. The proposed method provides an effective solution to deal with the multi-class imbalance problems. The experimental results show that the improved algorithm outperforms existing methods over a variety of datasets. It also demonstrates that the optimization of cost setup and ensemble learning are two critical factors to improve the cost sensitive learning.

## REFERENCES


[1] Q. Yang, X. Wu, "10 challenging problems in data mining research," Int'l J. Information Technology and Decision Making, vol. 5, pp.597–604, 2006.

[2] H. He, E.A. Garcia, "Learning from imbalanced data," IEEE Transactions on Knowledge and Data Engineering, vol. 21, pp.1263-1284, 2009.

[3] N.V. Chawla, A. Lazarevic, L. Hall, K. Bowyer, "SMOTEBoost: Improving prediction of the minority class in boosting," Proc. of 7th European Conf. Principles and Practice of Knowledge Discovery in Databases, pp. 107-119, 2003.

[4] T. Hoens, N.V. Chawla, "Generating Diverse Ensembles to Counter the Problem of Class Imbalance," Proc. of 14th Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 488-499, 2010.

[5] D.L. Wilson, "Asymptotic properties of nearest neighbor rules using edited data," IEEE Transactions on Systems, Man and Cybernetics, vol 30, pp. 408-421, 1972.

[6] V.S. Sheng, C.X. Ling, "Thresholding for making classifiers cost-sensitive," Proc. of 21th National Conference on Artificial Intelligence, pp. 476-481, 2006.

[7] Z. H. Zhou, X. Y. Liu, "Training cost-sensitive neural networks with methods addressing the class imbalance problem," IEEE Transactions on Knowledge and Data Engineering, vol. 18, pp. 63-77, 2006.

[8] P. Domingos, "MetaCost: A general method for making classifiers cost-sensitive," Proc. of ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 155–164, 1999.

[9] S. Wang, X. Yao, "Multiclass imbalance problems: Analysis and potential solutions," IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, vol. 42, pp. 1119-1130, 2012.

[10] Z.H. Zhou, J. Wu, W. Tang, "Ensembling neural networks: many could be better than all," Artificial intelligence, vol. 137, pp. 239-263, 2002.

[11] T. Ho, "The random subspace method for constructing decision forests," Pattern Analysis and Machine Intelligence, vol. 20, pp. 832–844, 1998.

[12] D. Martens, B. Baesens, T. Fawcett T, "Editorial Survey: Swarm Intelligence for Data Mining," Machine Learning, vol. 82, pp.1-42, 2011.

[13] D. Karaboga, B. Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony algorithm," Journal of Global Optimization, vol. 39, pp. 459–471, 2007.

[14] D. Karaboga, B. Akay, "A comparative study of artificial bee colony algorithm," Applied Mathematics and Computation, vol.214, pp.108–132, 2009.

[15] V.H. Jason, T.M. Khoshgoftaar, A. Napolitano, R. Wald, "Feature selection with high-dimensional imbalanced data," Proc. of IEEE International Conference on Data Mining Workshops:507-514 2009.