Selective Further Learning for Class Imbalanced Incremental Learning

Minlong Lin

Supervisor: Xin Yao and Ke Tang

***Abstract*—Incremental learning is a learning model for learning new information from new data. In incremental learning, the data sets are available chunk by chunk. The learning model should be able to learn new information from new data sets without accessing previously learned data sets and preserve previously learned information. In this report, class imbalance in incremental learning is considered and a new framework, Selective Further Learning (SFL), is proposed. SFL is an ensemble based method. In SFL, when a new data set becomes available, part of the current ensemble is selected to *Further Learn* the new data set. Some experiments on synthetic data sets are implemented and the results and analyses show that SFL is able to outperform a recently proposed method.**

Index *Terms*—Incremental Learning, Class Imbalance, Selective Further Learning

I. Problem Description

Normal machine learning problems require learning model to learning information from all the achieved data and all the data are stored. In practice, the data are usually updated all the time and new information is necessary to be learned from the new data. However, it is time consuming to learn new information with accessing to the previous data and sometimes storing the learned data is expensive. In this situation, the learning model is required to have the ability of learning new information from new data and at the same time preserving the previously learned information without accessing the previous data. This learning model is called incremental learning.

In incremental learning, the whole data set is not available in a lump. In another word, we can only get a part of the data set every time. We suppose that the whole data set S is divided into T subsets, i.e., $S_1, S_2, \dots, S_T$. The rules of S and S_t are denoted as R and R_t respectively.

In incremental learning, the aim of the learning model is to learn R by learning R_t from S_t respectively. The mainly difficulty is that the previous learned rules may be forgotten when the model learns new rules from new data subsets, especially when the rules of different data subsets are different. This phenomenon was called catastrophic forgetting. If $R_1 = R_2 = \cdots = R_T$, the learning model can learn R_1 form S_1 and R_1 will not be forgotten when new data subsets are learned. In this case, incremental learning is not real challenging. However, in practical, R_t are usually different between different data subsets, so the catastrophic forgetting may happen.

In our assumption, even though the rules are different between different data subsets, they are not inconsistent with R, i.e., the target concept is not changed. This phenomenon is also called virtual concept drift and it is different from real concept drift, in which the target concept is changed when new data subsets are available. In this research, it is assumed that no real concept drift will happen. Virtual concept drift was called sampling shift in [1] and it will be referred to in this work. When only sampling shift happens, if the rules can be characterized by a mean and covariance matrix, both of which are additive, then we can still get the rules of the whole data set easily. Unfortunately, most nonparametric approaches, such as Neural Networks, are not additive, and this comes into the challenge.

There is another issue in incremental learning, i.e., class imbalance. In normal learning model, class imbalance problem has been studied by many researchers and there are plenty of literatures addressing class imbalance problems [2], [3], [4]. Class imbalance problems can also occur in incremental learning. Class imbalance may occur in mainly two cases:

1) If the class distribution of the whole data set S is imbalanced, the class distribution of data subset S_T will also be imbalanced. Furthermore, it will be common that samples of the minority classes may be missing in some data subsets. Sampling shift will occur in this case and the learning model will also suffer from catastrophic forgetting.
2) Even though the class distribution of S is balanced, S_t may also be class imbalanced. In typical case, all of the partial sets S_t are class imbalanced but the combined data set S is class balanced. Sampling shift will also occur in this case.

In this research, we focused on class imbalance cases, in which sampling shift also occurred. Specifically, when sampling shift occurs, new classes may come up in the new data subset and some previous classes may loss in the new data subset. When class distribution of the whole data set is imbalanced, this phenomenon will be more likely to happen to the minority classes. This is also the main issue in this research.

The rest of this report is organized as follows. In Section II, we will briefly review some existed methods for incremental learning. In Section III, our method will be described. The experimental study will be presented in Section IV. Finally, we will conclude this research and discuss the future work in Section V.

The author and his supervisors are with the Nature Inspired Computation and Applications Laboratory, the Department of Computer Science and Technology, University of Science and Technology of China, Hefei, Anhui 230027, China. Xin Yao is also with CERCIA, the School of Computer Science, University of Birmingham, Edgbaston, Birmingham B15 2TT, U.K.

Emails:

Minlong Lin: sunnyboy@mail.ustc.edu.cn; Xin Yao: x.yao@cs.bham.ac.uk; Ke Tang: ketang@ustc.edu.cn.

II. Related Work

Some incremental learning methods have been proposed.

The Adaptive Resonance Theory modules map (ARTMAP) [5] was a kind of neural network architecture which can self-organize clusters in response to new data which are sufficiently different from previous ones. ARTMAP is capable of incremental learning and was frequently used in incremental learning [6], [7], [8]. ARTMAP can perform well in incremental learning. However, it is very sensitive to the vigilance parameter and it will cause over fitting if the vigilance parameter is not chosen properly.

Kasabov proposed a one-pass incremental learning algorithm, namely evolving fuzzy neural network (EFuNN) [9]. EFuNN can create new neurons for new input examples that are sufficiently different from previous ones. EFuNN can learn fast and adapt incrementally in an on-line mode. However, many parameters have to be set and the parameters have high influence in the training of EFuNN [10].

Both ARTMAP and EFuNN are methods for training single model. They learn new rules by changing the architecture of the models such as adding new neurons. Recently, some ensemble-based methods were proposed for incremental learning.

Learn++ [11] [12] is an ensemble method which was based on AdaBoost [13]. There are also some improved versions for Learn++, such as Learn++.MT [14], Learn++.MT2 [15], Learn++.NC [16] and Learn++.UDNC [17]. Learn++.MT and Learn++.NC was proposed for handling the problem of "outvoting" when learning new classes. Learn++.MT2 was proposed for handling the imbalance between partial data sets. Class imbalance in incremental learning was addressed in [17] and [18], where Learn++.UDNC and Learn++.SMOTE was proposed respectively. Learn++.UDNC was proposed for incrementally learning new information from class-imbalanced data sets. In its assumption, no real concept drift will happen, which is also the assumption in this research. However, Learn++.SMOTE was proposed for incrementally learning new information from class-imbalanced data sets in a non-stationary environments, i.e., real concept drift will happen.

In [19], Seipone and Bullinaria employed evolutionary neural networks to solve incremental learning. The evolutionary algorithm was used to evolve the training parameters for the neural networks, i.e., the number of hidden units, the learning rate, initial weight distributions, and so on. At the training process, each neural network is trained with one partial data set. However, the evolving process for optimizing the parameters considers the whole data set. Therefore, this approach is not strictly adapted to incremental learning.

Self-Organizing Neural Grove (SONG) [20] is also an ensemble method which uses Self-Generating Neural Networks (SGNNs) [21] as the individual learners. SGNN is based on Self-Organizing Map (SOM) and implemented as a self-generating neural tree (SGNT). A pruning method is introduced to prune the units of SGNT to reduce the computation time and the memory capacity. SONG has also been shown to be effective for incremental learning. Although the number of SGNTs is fixed, the number of total units of SGNTs grows as more data sets are incrementally learned. In [20], SONG was just employed to test its ability of incremental learning. Incremental learning was not the main issue and not deeply investigated.

Recently, methods based on Negative Correlation Learning (NCL) [22] have also been proposed for incremental learning [23], [24]. NCL is a method to construct neural networks ensemble. It is capable of improve the generalization performance of the ensemble by decreasing the error of every neural network and increasing the diversities between neural networks simultaneously. In [23], Fixed size NCL and Growing NCL was proposed. In Fixed size NCL, the size of the ensemble is fixed and all of the neural networks are trained when new data subsets become available. In Growing NCL, the size of the ensemble grows as the data sets are incrementally learned and only new added neural networks are trained when new data subsets become available. In [24], selective NCL was proposed. In selective NCL, new neural networks are added and trained when new data subsets become available and then a pruning method was employed to prune the ensemble to make the size of the ensemble fixed.

Although the existed methods have presented good performance on incremental learning, most of them did not consider the class imbalance in incremental learning. In this research, class imbalance in incremental learning will be considered as the main issue. According to the discussion before, Learn++.UDNC considered the same issue of this research. Therefore, we will mainly compare our method with Learn++.UDNC.

III. Our method

A. The Framework

In our previous work, i.e., Selective Negative Correlation Learning (SNCL) [24], selective ensemble was used. When new data subset became available, the previous ensemble was cloned and the new data subset was used to train the cloned ensemble. The two ensembles were combined to form an ensemble and half of the individuals in the ensemble were pruned to keep the size of the ensemble fixed. This method has the following two drawbacks. One is the training process. The new data subset is used to train the cloned ensemble. However, the cloned ensemble has been converged when training with the previous data set. It might be difficult for the cloned ensemble to learn new information from new data subset. The converging process might be slow. The other is the pruning process. When pruning the combined ensemble, the current data subset is used. If the rules of the current data subset are quite different from that of the previous data subset, i.e., sampling shift occurs, the pruning process will prune all of the individuals of the previous ensemble. This will cause catastrophic forgetting.

To overcome these drawbacks, we propose a new framework for incremental learning, i.e., Selective Further Learning (SFL). The following presents the process of the framework (when new data subset becomes available), where the number of the individuals in the previously trained ensemble will be denoted as M.

step 1. Fetch $\lfloor M/2 \rfloor$ individuals from the current ensemble and combine them as an ensemble ens_{fet}, i.e.,

the current ensemble is divided into two ensemble: ens_{fet} and ens_{res};

step 2. Replace one individual of ens_{fet} by a new initialized one;

step 3. Train ens_{fet} with the new data subset;

step 4. Combine ens_{fet} and ens_{res}: $ens_{new} = ens_{fet} \cup ens_{res}$;

In this framework, no pruning process will be executed so that the risk of catastrophic forgetting is reduced. On the other hand, the replacing process in step 2 will make the ensemble easier to learn new information.

B. The Structure of the Ensemble

We used Multi-Layer Perceptrons (MLP) to construct the ensemble. For a data set with c classes, the number of output nodes of each MLP was set as c. For a testing example, the output of ith output node indicates the possibility that the example belongs to class i. In incremental learning mode, the data sets are available chunk by chunk. Different chunks might contain different classes. At testing stage, when an example of class i is used for testing, the outputs of the MLPs which have not trained by any example of class i may mislead the outputs of the ensemble. In this situation, some modifications were made to the structure of MLP.

An extra weight was added to every output node in every MLP. The weight was initialized as 0 at the initial stage and updated during learning every new data subset. We suppose the number of examples of class i in the new data subset is n_i. After training ens_{fet}, the extra weight of the ith output node was updated by n_i. The accumulation was executed for all the output nodes of the MLPs in ens_{fet}. At the testing stage, the output of the ensemble was calculated by the weighted average of MLPs:

$$o_i = \sum_{k=1}^{M} w_{ik} o_{ik} / \sum_{k=1}^{M} w_{ik}, i = 1,2, \dots, c, \quad (1)$$

where w_{ik} is the extra weight of the ith output node of the kth MLP, o_{ik} is the output of the ith output node of the kth MLP and o_i is the ith output of the ensemble. Equation (1) was used only at the testing stage. At the training stage, the output of the ensemble was calculated by the arithmetical average of MLPs.

On the other hand, when new classes appear in the new data subset, new output nodes were added to all of the MLPs in the current ensemble. The connection weights to the hidden nodes were set by initialization method and the extra weights were initialized as 0.

C. Fetching process

The fetching process in step 1 of the framework will base on the current data subset S_t. The individuals will be added to ens_{fet} one by one. Every time, the individual which makes ens_{fet} perform the worst on the current data subset will be added to ens_{fet}. At the same time, the following constraint must be satisfied. If the current data subset do not contain some classes that have appeared in the previous data subset, the fetching process should ensure that at least one MLP that has been trained with the data of those classes should not be added to ens_{fet}. To formulate this constraint, after dividing the current ensemble into ens_{fet} and ens_{res}, the following constraint should be satisfied in ens_{res}:

$$\prod_{i \in L} (\sum_k w_{ik}) \neq 0. \quad (2)$$

where $L = \{i | \text{class } i \text{ is not contained in } S_t\}$. If there is no MLP that can be added to ens_{fet}, a new initialized MLP will be generated and added to ens_{fet}.

D. Training process

We have proposed a Dynamic Sampling (DyS) method for class imbalance problems [25], which can be used as the training method in step 3 of the framework. The main process of DyS for an ensemble is presented as follows (in one epoch):

step 1. Randomly fetch an example **x** from the training set;

step 2. Estimate the probability p that the example should be used for updating the ensemble.

step 3. Generate a uniform random real number μ between 0 and 1.

step 4. If $\mu < p$, then use **x** to update the ensemble using Negative Correlation Learning (NCL) [22].

step 5. Repeat steps 1 to 3 until there is no example in the training set.

The above steps will be repeated until stop criterion is satisfied. The following shows the method for estimating p, which was the main issue in DyS.

For an example belonging to class i, the real output of the example is denoted as $\mathbf{y} = \{y_k | k = 1,2, \dots, c\}$. The margin can be defined as

$$\delta = y_i - \max_{k \neq i} \{y_k\}. \quad (3)$$

δ indicates the confidence of the MLP to classify the example as class i and $\delta < 0$ indicates that the example is misclassified by the current MLP. The probability that the example should be used to update the MLP should be negatively correlated to δ.

In addition to the margin, the imbalance should be considered. The example belonging to a minority class should have a higher probability to be used to update the MLP than that of a majority class. The ratio of class i is defined as $r_i = n_i / n$, where n_i is the number of examples belonging to class i and n is the number of all the examples. The probability that the example will be used to update the MLP should be negatively correlated to r_i.

In class imbalance learning, the margin that the MLP distinguishes an example belonging to class i from other classes is defined as

$$\Delta = \delta \cdot r_i / z, \quad (4)$$

where z is a normalization factor and we set $z = \min_k \{r_k\}$ so that the margin for the class with the lowest number of examples is equal to (3).

$\Delta < 0$ indicates that the example is misclassified and the example need to be learned. So $p = 1$ if $\Delta < 0$. Otherwise, $\Delta \in [0, \max_j \{r_j\} / \min_j \{r_j\}]$, and the probability p should be negatively correlated to Δ, so the probability can be estimated as $p = \exp\{-\Delta\}$. Therefore, the probability that an example belonging to class i will be used to update the MLP can be heuristically estimated as

$$p = \begin{cases} 1, if\ \delta < 0 \\ \exp(-\delta \cdot r_i / \min_k\{r_k\}), otherwise\ . \end{cases} \quad (5)$$

In this selection mechanism:

i. The examples that are misclassified will be selected to update the MLP.
ii. For the examples that are correctly classified, the examples of minority classes are emphasized more than those of majority classes.
iii. If the examples are correctly classified, those which are classified more ambiguous (i.e., the value of δ in (3) is smaller) are emphasized more.

The selection mechanism can allay the affection of class imbalance. However, if the selection mechanism is used over the training set directly, due to the class imbalance of the training set, more examples of the majority classes will be tested whether be used to update the MLP or not according to (5). Especially, at the beginning of the training process, the initial MLP will misclassify about 50% of the examples for all the classes. According to the first condition of (5), the misclassified examples will be used for training. Therefore, the selected examples will still be imbalanced and the MLP may easily be biased towards the majority classes.

To avoid the bias, an oversampling process will be implemented at the beginning of every epoch. At the beginning of the first epoch, the examples of all classes except for the largest class will be duplicated to make the data set balanced, i.e., the number of every class is equal to that of the largest class. As the training process goes on, the duplicate ratio will be attenuated. Denote the duplicate ratio for class i at the first epoch as α_i, the duplicate ratio for class i will be heuristically attenuated to $\alpha_i / \ln ep$ at the end of the epth epoch, where $ep > 2$.

IV. Experimental Study

A. Experimental Setup

For comparing with Learn++.UDNC, the synthetic data set which was used in [17] will be used in the experimental study. In [17], the data set was generated by 2D Gaussian distribution with predefined means and covariance matrices for the four classes. The means were $\mu_1 = (0,0)$, $\mu_2 = (1,1)$, $\mu_3 = (-1,0)$ and $\mu_4 = (1,-1)$. The features were assumed to be uncorrelated and the variances were $\sigma_1 = 0.15$ and $\sigma_{2,3,4} = 0.35$. TABLE I shows the data distributions for two types of data sets. It can be observed that the training sets are class imbalanced and there are new classes in new data subsets. Especially, in Type B, class 3 appeared in S_3 and then disappeared in S_4.

One MLP (i.e., $M = 1$) was used at the beginning, i.e., when the first data subset became available. The number of hidden nodes in every MLP was set to 20 and the activation functions of hidden nodes and output nodes were both sigmoid functions. The training error goal was 0.05 and the number of maximal epochs was 100. The data generation was repeated 30 times and in every time, the training process was executed one time. The results were the average on the 30 executions.

B. Experimental Results

The recall of every class was used as the metric. After training with every data subset, the recalls of the classes on testing data set were estimated. TABLE II presents the results comparing with Learn++.UDNC, the results of which were directly obtained from [17]. In TABLE II, AVG denotes the arithmetical average of the recalls of all the classes, Im denotes the improvement from the first time the class appears to the last data subset. For example, class 2 was first appeared in S_2, so the improvement was the recall value after the training of S_4 minus the recall value after the training of S_2. In the AVG column, the improvement was the last AVG value minus the first AVG value.

TABLE I Two types of data distributions of the synthetic data sets, where C_i denotes class i, S_i denotes the ith data subset.

	Type A				Type B			
	C_1	C_2	C_3	C_4	C_1	C_2	C_3	C_4
S_1	10	0	0	500	10	0	0	500
S_2	10	500	0	500	10	500	0	500
S_3	10	500	500	500	10	500	500	500
S_4	10	500	10	500	10	500	0	500
Test	200	200	200	200	200	200	200	200

It can be observed from TABLE II that, SFL can make more improvement than Learn++.UDNC. SFL is capable of handling class imbalance. After the training of S_4, the recalls of class 1, which is a minority class, are higher than that of Learn++.UDNC in both Type A and Type B. SFL can also deal with new classes that come up in the new data subset (class 2 in S_2 and class 3 in S_3). In Type B, class 3 disappears in S_4. However, after training with S_4, the recall of class 3 was not reduced. Therefore, when previous classes disappear in the new data subset, SFL can also preserve the learned information. Although SFL did not perform better than Learn++.UDNC when comparing by the value of average recall in Type A, SFL used much less MLPs. After training with S_4, 2 MLPs were used in SFL while 60 MLPs were used in Learn++.UDNC.

C. Further Analyses

To see the performance of SFL with more MLPs in the ensemble, another experiment with $M = 10$ was executed. The results are presented in TABLE III. It can be observed from TABLE III that when M increase, the performance of SFL become worse. Especially, new classes were not well learned after their first came up. To find out the reason of this phenomenon, some more detailed results were analyzed. After the training with every subset, the recalls of all the classes of every single MLP on both training set and testing set were recorded. The following presents some analysis.

- According to comparing of the results on training set and testing set, we concluded that over fitting did not occur.
- After training with S_3, in which class 3 came up, the MLPs in ens_{fet} can get a good recall on class 3. However, when ens_{fet} and ens_{res} was combined, the recall value of class 3 degrade significantly. This is because class 3 has not been learned by ens_{res} so that ens_{res} will misclassify class 3 to other classes. When ens_{fet} and ens_{res} was combined, ens_{res} may mislead the classification of an example of class 3.

TABLE II THE RECALLS (%) OF EVERY CLASS ON TESTING SET, WHERE AVG DENOTES THE ARITHMETICAL AVERAGE OF THE RECALLS OF ALL THE CLASSES, Im DENOTES THE IMPROVEMENT FROM THE FIRST TIME THE CLASS APPEARS TO THE LAST DATA SUBSET

Type A

		C_1	C_2	C_3	C_4	AVG
Learn++.UDNC	S_1	91.55±2.25	0	0	98.40±0.55	47.49±0.50
	S_2	78.65±3.53	91.55±1.48	0	98.90±0.33	67.28±0.96
	S_3	59.15±4.33	93.70±0.93	90.30±1.63	98.60±0.37	85.44±1.00
	S_4	56.75±4.56	94.75±0.57	90.35±1.61	98.75±0.35	85.15±1.07
	Im	-34.80	3.20	0.05	0.35	37.66
SFL($M=1$)	S_1	98.35±1.50	0	0	82.23±5.52	45.15±1.16
	S_2	97.30±2.20	67.78±9.03	0	83.02±4.93	62.03±2.68
	S_3	89.15±5.44	85.43±4.09	56.18±18.11	85.40±4.33	79.04±4.07
	S_4	74.13±10.63	88.08±2.97	79.90±9.45	87.28±4.15	82.35±1.60
	Im	-24.22	20.3	23.72	5.05	37.2

Type B

		C_1	C_2	C_3	C_4	AVG
Learn++.UDNC	S_1	91.45±2.90	0	0	95.35±1.12	46.70±0.49
	S_2	79.05±3.85	89.70±1.65	0	96.20±0.76	66.24±0.87
	S_3	52.80±4.14	92.10±1.14	91.40±1.13	95.65±0.51	82.99±1.08
	S_4	50.30±3.86	93.25±1.06	91.15±1.14	95.75±0.51	82.61±1.01
	Im	-41.15	3.55	-0.25	0.40	35.91
SFL($M=1$)	S_1	98.37±1.37	0	0	83.15±5.60	45.38±1.20
	S_2	97.33±1.30	67.00±12.59	0	84.58±3.83	62.23±3.10
	S_3	89.35±4.91	85.47±4.11	58.50±14.78	86.40±2.96	79.93±2.93
	S_4	89.12±6.49	85.65±4.36	59.87±15.17	85.78±2.78	80.10±3.02
	Im	-9.25	18.65	1.37	2.63	34.72

TABLE III THE RECALLS (%) OF EVERY CLASS ON TESTING SET OF SFL WITH $M=10$, WHERE AVG DENOTES THE ARITHMETICAL AVERAGE OF THE RECALLS OF ALL THE CLASSES, Im DENOTES THE IMPROVEMENT FROM THE FIRST TIME THE CLASS APPEARS TO THE LAST DATA SUBSET

Type A

		C_1	C_2	C_3	C_4	AVG
SFL ($M=10$)	S_1	98.90±1.04	0.00±0.00	0.00±0.00	78.98±4.80	44.47±1.04
	S_2	96.75±2.71	56.43±11.28	0.00±0.00	84.05±3.36	59.31±2.72
	S_3	92.48±4.68	79.77±5.13	29.47±21.43	82.97±3.87	71.17±5.53
	S_4	84.40±8.29	74.15±9.05	57.18±24.64	89.02±3.13	76.19±5.12
	Im	-14.5	17.72	27.71	10.04	31.72

Type B

		C_1	C_2	C_3	C_4	AVG
SFL ($M=10$)	S_1	98.43±1.33	0	0	82.45±4.47	45.22±0.91
	S_2	96.15±2.65	55.15±11.34	0	86.68±3.14	59.50±2.81
	S_3	93.00±4.08	80.18±4.94	29.60±22.74	84.63±3.46	71.85±5.66
	S_4	91.10±3.86	75.13±6.05	22.77±27.33	89.67±2.34	69.67±6.68
	Im	-7.33	19.98	-6.83	7.22	24.45

- Besides, if the MLPs of ens_{fet} did not perform well at the overall level (e.g., the recalls are high on most of the classes but 0 on one class), they may also mislead the classification. Therefore, when updating the extra weights for every MLP, the overall performance on all classes should be considered.

According to the above analysis, some modifications were made to SFL. The modifications were mainly on step 3 of the framework in Section III.A, including the updating of the extra weights in Section III.B and the training process in Section III.D.

1) After training ens_{fet} with new data subset, the extra weight of the ith output node in the mth MLP was updated by $n_i g_m$, where n_i is the number of examples of class i and g_m is the geometric mean of all the recalls of the mth MLP on current data subset. At the same time, the training process should ensure that not all the g_m values are 0.
2) If there are new classes in the new data subset, the extra weights in ens_{res} was also updated. Denote w_{mi} as the extra weight of the ith output node in the mth MLP

TABLE IV THE RECALLS (%) OF EVERY CLASS ON TESTING SET OF SFL_{NEW} WITH $M = 10$, WHERE AVG DENOTES THE ARITHMETICAL AVERAGE OF THE RECALLS OF ALL THE CLASSES, Im DENOTES THE IMPROVEMENT FROM THE FIRST TIME THE CLASS APPEARS TO THE LAST DATA SUBSET

Type A

		C_1	C_2	C_3	C_4	AVG
SFL_{new} ($M = 10$)	S_1	97.65±1.64	0	0	85.30±4.24	45.74±0.89
	S_2	88.97±4.49	89.92±3.53	0	78.70±6.73	64.40±1.96
	S_3	71.92±5.49	91.22±3.31	80.85±8.11	82.65±7.48	81.66±2.73
	S_4	67.25±2.77	91.62±3.11	88.00±3.82	85.03±6.02	82.98±1.58
	Im	-30.4	1.7	7.15	-0.27	37.24

Type B

		C_1	C_2	C_3	C_4	AVG
SFL_{new} ($M = 10$)	S_1	97.88±1.53	0.00±0.00	0.00±0.00	85.12±4.30	45.75±0.89
	S_2	90.48±3.47	86.70±6.20	0.00±0.00	78.18±7.17	63.84±2.24
	S_3	70.80±9.85	90.35±3.67	81.00±9.39	82.70±6.19	81.21±1.98
	S_4	77.60±6.53	90.53±3.16	74.83±7.64	82.57±6.79	81.38±1.59
	Im	-20.28	3.83	-6.17	-2.55	35.63

in ens_{res}, it was updated as $w_{mi}\theta_m$, where θ_m is the average recall of the mth MLP on the new data subset.

3) After training ens_{fet} with new data subset, ens_{fet} and ens_{res} were combined to form a temp ensemble ens_{tmp}. If the geometric mean of all the recalls of ens_{tmp} on the new data subset was less than a predefined threshold (e.g. o.5), then redo the training process.

The results of SFL with the above modifications (denoted by SFL_{new}) are presented in TABLE IV ($M = 10$). It can be observed from TABLE IV that SFL_{new} performs better than SFL. Comparing with the results of Learn++.UDNC in TABLE II, SFL_{new} shows more balance performance on all the classes. The number of MLPs that used in SFL_{new} are less than that used in Learn++.UDNC. Furthermore, in Learn++.UDNC, numbers of individual classifiers (e.g., MLPs) will be added to the ensemble as more data sets become available while in SFL_{new}, the size of the ensemble will be almost fixed.

V. CONCLUSION AND FUTURE WORK

In this research, the main issues in incremental learning were discusses and class imbalance in incremental learning was focused on. According to the discussing the drawbacks of our previously proposed method, an ensemble base method, named Selective Further Learning (SFL) was proposed. In SFL, when a new data set becomes available, part of the current ensemble is selected to training with the new data set. In order to address potential variety on classes in the new data set, i.e., new coming up classes or disappearing of previous classes, an extra weight is added to every output node of every MLP in the ensemble. The extra weight indicates the confidence that the MLP has learned about the relevant class. On the other hand, Dynamic Sampling (DyS) method which was proposed for class imbalance problems was used as the training method to handle the class imbalance in incremental learning.

In the experimental study, we compared SFL with a recently proposed method, i.e., Learn++.UDNC. The results have shown that SFL can outperform Learn++.UDNC. Some further analyses were also presented to find out the detail behaviors of SFL and some modifications were made to improve SFL.

The experimental results and discussions show that the framework of SFL is efficient for class imbalanced incremental learning. However, some details of the framework, such as the training method and the ensemble combining method, can be designed better to improve SFL. This will be investigated in our future work.

ACKNOWLEDGEMENT

This research has been partially supported by IEEE CIS Walter Karplus Summer Research Grant 2011.

REFERENCES

[1] M. Salganicoff, "Tolerating concept and sampling shift in lazy learning using prediction error context switching," *Artificial Intelligence Review*, vol. 11, pp. 133-155, 1997.

[2] N. Japkowicz and S. Stephen, "The class imbalance problem: A systematic study," *Intelligent Data Analysis*, vol. 6, pp. 429-449, 2002.

[3] N. V. Chawla, N. Japkowicz and A. Kotcz, "Editorial: special issue on learning from imbalanced data sets," *ACM SIGKDD Explorations Newsletter*, vol. 6, pp. 1-6, 2004.

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

[5] G. A. Carpenter, S. Grossberg and J. H. Reynolds, "ARTMAP: Supervised real-time learning and classification of nonstationary data by a self-organizing neural network," *Neural networks*, vol. 4, pp. 565-588, 1991.

[6] G. A. Carpenter, S. Grossberg, N. Markuzon, J. H. Reynolds, and D. B. Rosen, "Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps," *IEEE Transactions on Neural Networks*, vol. 3, pp. 698-713, 1992.

[7] J. R. Williamson, "Gaussian ARTMAP: A neural network for fast incremental learning of noisy multidimensional maps," *Neural Networks*, vol. 9, pp. 881-897, 1996.

[8] M. C. Su, J. Lee and K. L. Hsieh, "A new ARTMAP-based neural network for incremental learning," *Neurocomputing*, vol. 69, pp. 2284-2300, 2006.

[9] N. Kasabov, "Evolving fuzzy neural networks for supervised/unsupervised online knowledge-based learning.," *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics: a publication of the IEEE Systems, Man, and Cybernetics Society*, vol. 31, p. 902, 2001.

[10] C. Zanchettin, L. L. Minku and T. B. Ludermir, "Design of experiments in neuro-fuzzy systems," *International Journal of Computational Intelligence and Applications*, vol. 9, pp. 137-152, 2010.
[11] R. Polikar, L. Udpa, S. S. Udpa, and V. Honavar, "Learn++: An Incremental Learning Algorithm for Supervised Neural Networks," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 31, p. 497, 2001.
[12] R. Polikar, J. Byorick, S. Krause, A. Marino, and M. Moreton, "Learn++: a classifier independent incremental learning algorithm for supervised neural networks," in *International Joint Conference on Neural Networks*, pp. 1742 – 1747, 2002.
[13] Y. Freund and R. Schapire, "A short introduction to boosting," *Journal of Japanese Society for Artificial Intelligence*, vol. 14, pp. 771-780, 1999.
[14] M. Muhlbaier, A. Topalis and R. Polikar, "Learn++. mt: A new approach to incremental learning," *Multiple Classifier Systems*, pp. 52-61, 2004.
[15] M. Muhlbaier, A. Topalis and R. Polikar, "Incremental learning from unbalanced data," in *IEEE International Joint Conference on Neural Networks*, pp. 1057-1062, 2004.
[16] M. D. Muhlbaier, A. Topalis and R. Polikar, "Learn++. NC: combining ensemble of classifiers with dynamically weighted consult-and-vote for efficient incremental learning of new classes," *IEEE transactions on neural networks*, vol. 20, pp. 152-168, 2009.
[17] G. Ditzler, M. Muhlbaier and R. Polikar, "Incremental Learning of New Classes in Unbalanced Datasets: Learn++. UDNC," *Multiple Classifier Systems*, pp. 33-42, 2010.
[18] G. Ditzler, R. Polikar and N. Chawla, "An Incremental Learning Algorithm for Non-Stationary Environments and Class Imbalance," in *International Conference on Pattern Recognition*, pp. 2997-3000, 2010.
[19] T. Seipone and J. A. Bullinaria, "Evolving improved incremental learning schemes for neural network systems," in *IEEE Congress on Evolutionary Computation*, pp. 2002-2009, 2005.
[20] H. Inoue and H. Narihisa, "Self-organizing neural grove and its applications," in *IEEE International Joint Conference on Neural Networks*, pp. 1205-1210, 2005.
[21] W. X. Wen, H. Liu and A. Jennings, "Self-generating neural networks," in *International Joint Conference on Neural Networks*, pp. 850-855, 1992.
[22] Y. Liu and X. Yao, "Simultaneous training of negatively correlated neural networks in an ensemble," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 29, pp. 716-725, 2002.
[23] F. L. Minku, H. Inoue and X. Yao, "Negative correlation in incremental learning," *Natural Computing*, vol. 8, pp. 289-320, 2009.
[24] K. Tang, M. Lin, F. L. Minku, and X. Yao, "Selective negative correlation learning approach to incremental learning," *Neurocomputing*, vol. 72, pp. 2796-2805, 2009.
[25] M. Lin, K. Tang and X. Yao, "A Dynamic Sampling Approach to Training Neural Networks for Multi-class Imbalance Classification," submitted to *Pattern Recognition*.