Incremental SampleBoost for Efficient Learning from Multi-Class Data Sets

Mohamed Abouelenien* Xiaohui Yuan*

Abstract

Ensemble methods have been used for incremental learning. Yet, there are several issues that require attention, including elongated training time and smooth integration of new examples. In this article, we introduce an incremental SampleBoost method that learns efficiently from new data by employing a class-based down sampling strategy with an error parameter. Our novel weight initialization scheme integrates new information effectively with the retained important examples. Experimental results shows the superiority of our method compared to state-of-the-art incremental and batch methods including AdaBoost, AdaBoost.M1, SAMME, and Learn++.

Keywords: Incremental Learning, Ensemble Classification, Sampling, Weighting

1 Introduction

Massive amount of data is generated in many applications and specially in the medical field. It is common in such cases to continuously acquire data that contain important information and representative examples. An example of this is Capsule Endoscopy (CE) technology. CE is used to visualize the entire small intestine in order to detect abnormality. A patient swallows a tiny capsule-shaped pill that has a camera. The camera records a video of up to 8 hours resulting in thousands of images. CE is very successful in visualizing bleeding, tumors, and other types of abnormality compared to conventional technologies such as colonoscopy and esophagogastroduodenoscopy. However, thousands of images are generated daily from such technologies. Hence, the need arises for learning approaches that efficiently update the existing models with new examples. There are two strategies to employ newly acquired data: catastrophic forgetting [6], i.e., discarding the existing model and starting from scratch, and integrating knowledge in the new data set to the model.

Boosting methods combines weak classifiers to deal with difficult problems. Since weak classifiers are trained sequentially, there is great potential to perform incremental learning [10]. However, the data size becomes a difficult challenge in the application of a boosting method due to multiple trainings to complete an ensemble. In addition, boosting suffers from early termination when repetitively misclassified examples exist [1]. Such over weighted misclassified examples inflate the total error and cause termination of the training process. Imbalanced data poses another challenge. The improvement of performance of the minority classes is limited since the decision boundary is biased to favor the majority classes.

SampleBoost was developed to destabilize weak classifiers and has demonstrated improved accuracy and efficiency [1]. In this paper we extend our SampleBoost to learn incrementally from new data streams for both balanced and imbalanced datasets. An intelligent integration process is employed by assigning the new samples proper weights. A weighted selection process is then applied to create a representative down sampled training set that achieves improved efficiency and accuracy. Employing a selection mechanism combined with an error parameter that adjusts the error bound, Incremental SampleBoost (ISB) is able to avoid early termination, learn efficiently for large multi-class datasets, and allow an effective transition when new data becomes available. Moreover, the method avoids favoring majority classes by effectively balancing the classes even in multi-class datasets.

In our experiments we compare our ISB method with AdaBoost.M1 [4], SAMME [15], and Learn++ [10] methods in batch learning and incremental learning settings [7]. In addition to a Gaussian synthetic data set, we evaluate the methods with the UCI image segmentation data set and Capsule Endoscopy videos.

The rest of the paper is organized as follows. In Section 2, we present the state-of-the-art incremental learning methods. In Section 3, we present our Incremental SampleBoost method. Section 4 discusses our experimental results. Our concluding remarks are presented in Section 5.

2 Related Work

Support Vector Machines (SVM) have been improved to learn incrementally from new data [3, 11]. Syed et

*Computer Vision and Intelligent Systems Lab, Computer Science and Engineering Departement, University of North Texas. {mohamed, xiaohui.yuan}@unt.edu

al. [12] retains the support vectors of each step to preserve the boundary information gained. Giritharan et al. [5] developed geometric incremental SVM that identifies the skin of convex hull and integrates new examples to update models. Ozawa et al. [9] introduced incremental Principal Component Analysis that combines dimensionality reduction and classification. Zhou and Chen [14] embedded feed-forward neural networks in the leaves of a decision tree that grows to accommodate new examples. Utgoff et al. [13] employed a virtual pruning for incremental decision tree induction.

Extension of ensemble has also been attempted to adapt to new examples. AdaBoost.M1 was modified to integrate new examples [7]. The weights of a batch of training examples are initialized and a weak hypothesis is iteratively generated from these samples for a predetermined number of iterations. After each iteration the samples weight distribution is adjusted and a weight α is assigned to each weak hypothesis based on the evaluation of the training data. Once a new batch of samples is available, their weights are initialized equally according to the size of the new batch. The new batch is combined with the old set and thereby resulting in a larger training set. After training all batches, the weak hypothesis are combined using weighted majority voting. Following this idea, we extend a multi-class boosting method SAMME for incremental learning.

Polikar et al. [10] introduced Learn++ based on the framework of AdaBoost. This method was then extended in Learn++.NC to integrate new classes [8]. In Learn++, the training data is divided into two halves: TR_t is used to train the weak classifier; while both TR_t and TE_t combined are used for evaluation. The two subsets ensure that misclassified samples have higher probability of being included in TR_t. When a new batch of samples is ready, the samples weights are are equally initialized according to the size of the new batch. The evaluation process is performed using the current ensemble. At each iteration, the weighted error is calculated after evaluation and the weak hypothesis weight is assigned. A composite hypothesis is created using weighted majority voting of all current t iterations. The training set is re-evaluated using the composite hypothesis, and accordingly a composite weighted error is calculated. A weight is thereby assigned to the composite hypothesis. The final hypothesis is created using weighted majority voting on all composite hypothesis.

3 Method

3.1 Boosting Weighted Error Analysis Our analysis shows that repeatedly misclassified samples increases the weighted error of the boosting methods rapidly which results in zero-weighted hypothesis. When this occurs, some algorithms disregard the trained iteration and continue while others terminate the learning process. We provide a general analysis that is applicable to AdaBoost, AdaBoost.M1, and SAMME boosting methods. During each training iteration, the boosting method trains a weak classifier. Decisions of a series of weak hypothesis are then combined into a strong one to minimize the generalization error. Each weak hypothesis is evaluated using the training data in each iteration. Based on this evaluation, each weak hypothesis is assigned a weight α that shows the strength of its contribution to the overall decision. The evaluation assigns higher weights for misclassified samples and lower weights for correctly classified ones to alter the data distribution and focus on hard to classify samples.

However, repetition of misclassified samples results in extreme weight increase. This increase forces the weighted error above its maximum bound and accordingly assign a zero weight for the corresponding weak hypothesis as early as the second iteration. The mechanism of boosting methods allows adding classifiers that can construct a modified decision boundary. However, when these weak classifiers produce the same misclassified samples, the boosting method terminates and converges to a single classifier that requires larger training time. The analysis shows that if any number of samples is repeatedly misclassified, the boosting methods generates zero-weighted hypothesis and terminates rapidly.

Given a training set $(x_1, y_1), \ldots, (x_N, y_N)$, where $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y} = \{1, \ldots, K\}$. N is the total number of examples and K is the total number of classes. Each example has an initial weight $w_1(i) = 1/N$.

We assume that m samples are misclassified after training the first iteration, where $m \in \{1, \ldots, N\}$. Following AdaBoost.M1 and SAMME algorithms, the weighted error is calculated according to

$$\epsilon_t = \sum_{i=1}^{N} w_t(i) [\![y_i \neq h_t(x_i)]\!] \tag{3.1}$$

$[\![\cdot]\!]$ denotes an indicator function that returns 1 if true and 0 otherwise. $h_t(x_i)$ represents the class label decided by the weak hypothesis of iteration t. The weighted error of the first iteration is then equal to $\epsilon_1 = \frac{m}{N}$. After training the second iteration, assume that same m samples are misclassified. The weights of the m misclassified samples will be adjusted to

$$w_2 = \frac{e^{\alpha_1}}{NW_2}$$

and for the $N - m$ correctly classified samples to

$$w_2 = \frac{e^{-\alpha_1}}{NW_2}$$

W_2 represents a normalization factor. $W_2 = \sum_1^N w_2 = \frac{m}{N}e^{\alpha_1} + \frac{N-m}{N}e^{-\alpha_1}$ and α represents the contribution weight of the weak hypothesis.

$$\alpha_1 = \frac{1}{2}\log(\frac{1-\epsilon_1}{\epsilon_1}) + \beta \tag{3.2}$$

β represents a factor that is equal to 0 for AdaBoost and AdaBoost.M1, and to $\frac{1}{2}\log(K-1)$ for SAMME algorithm. On repetition of misclassified samples, the weighted error following the second iteration is adjusted to

$$\epsilon_2 = \frac{\frac{m}{N}e^{\alpha_1}}{\frac{m}{N}e^{\alpha_1} + e^{-\alpha_1} - \frac{m}{N}e^{-\alpha_1}} = \frac{e^{2\alpha_1}}{e^{2\alpha_1} + \frac{N}{m} - 1} \tag{3.3}$$

The weighted error can be simplified after substitution with the values of ϵ_1 and α_1 to

$$\epsilon_2 = \frac{[\frac{N}{m} - 1][e^{2\beta}]}{[\frac{N}{m} - 1][e^{2\beta}] + \frac{N}{m} - 1} \tag{3.4}$$

The weighted error can be further simplified to

$$\epsilon_2 = 1 - \frac{1}{e^{2\beta} + 1} \tag{3.5}$$

Substituting with the the value of β, we get

$$\epsilon_2 = \begin{cases} 0.5 & \text{AdaBoost} \\ 0.5 & \text{AdaBoost.M1} \\ 1 - \frac{1}{K} & \text{SAMME} \end{cases} \tag{3.6}$$

In all boosting methods, the weighted error sharply increases and passes the upper bound. This results in a zero weight for the weak hypothesis due to repetition of misclassified samples.

3.2 Incremental SampleBoost Incremental SampleBoost, presented in Algorithm 1, employs a class-based weighted sampling. In each training round, a number of examples is selected from each class. The selection process in only dependent on the distribution within each class. Hence, the data distribution decides the selection probability. For each class, highly weighted examples have greater probability of being selected. After each training iteration, the entire training set is evaluated and the class-based weighted selection process is repeated.

When a new batch of examples is available, their weights are initialized following our Algorithm 2. The initialization process plays a vital role in reducing the generalization error and allowing smooth transition when aggregating new data. The training set is evaluated and the same selection process is repeated during every round according to the weights of the now larger training set. In balanced training, the number of selected examples per class can be fixed during iteration. In case of imbalanced, multi-class problem, especially with a high imbalance ratio, the size can be determined based on the size of the smallest minority class. Assume $|s_l|$ is the size of the smallest class. A set of $|s_a|$ samples should be selected from all the other classes using our selection mechanism so that $|s_l| = |s_a|$. The selected samples set then is

$$S_t = \{\cup s_{l_q}, \cup s_{a_z}\}, \;\; where \;\; S_t \subset R_t \tag{3.7}$$

S_t represents the selected samples set upon which the training will occur and R_t represents the entire training set. s_{l_q} is the samples present in q number of minority classes, while s_{a_z} is the samples present in z number of classes other than minority. This sampling process is aimed to recover balance among all classes. Also, it allows flexibility of training class size when new examples become available for the minority class.

In contrast to Learn++, to reduce evaluation time during an iterative training process, and to allow smooth transition on the introduction of new data, the evaluation process is done with only the corresponding hypothesis. In learn++, using ensemble hypothesis will result in a significant increase in training time as well as in the weighted error as it will be biased towards old data. Based on the evaluation of each hypothesis, a weight α is assigned.

We include an error parameter γ to the loss function. The hypothesis weight will be calculated according to Equation 3.8. Hence, the maximum weighted error condition will be $\frac{\gamma}{1+\gamma}$. If the weighted error exceeds this value, the weak hypothesis does not contribute to the overall decision, i.e., $\alpha_t = 0$.

Whenever new data is available, it is combined with the older samples for the selection process. For incremental learning, only the samples weights of the last training iteration in addition to the final hypothesis are required from previous knowledge.

We assume a training set formed of batches of data $\mathcal{D}_p$, where $p = 1, \ldots, P$. Each batch is formed of samples $(x_1, y_1), \ldots, (x_{N'}, y_{N'})$ where $x_i \in \mathcal{X}$, $y_i \in \mathcal{Y} = \{1, \ldots, K\}$, and total number of samples $N = PN'$. In the description of this algorithm, we use $[\![\cdot]\!]$ to denote the indicator function that returns 1 if true and 0 otherwise and use $\mathcal{I}[.]$ to denote the indicator function that returns 1 if true and -1 otherwise.

3.3 Weight Initialization Weight initialization for the new examples plays a very important role in successfully learning the updated information. Assigning maximum weights results in driving the focus of a weak learner entirely on the new data and loses generaliza-

Algorithm 1 Incremental SampleBoost

for $p = 1, \ldots, P$: **do**
  Initialize distribution weights of new batch according to Algorithm 2.
  for $t = 1, \ldots, T$: **do**
    Set training data $R_t = \{\mathcal{D}_1, \ldots, \mathcal{D}_p\}$
    Select a subset $S_t \subset R_t$, $S_t = \{\cup s_j\}$, $j = \{1, \ldots, K\}$ according to distribution. In case of imbalanced training S_t is calculated according to Equation 3.7
    Train a weak learner h_t using S_t.
    Evaluate the weak learner using R_t.

$$h_t = \arg\min_{h_t \in H} \epsilon_t = \sum_{i=1}^{|R_t|} w_t(i) [\![y_i \neq h_t(x_i)]\!]$$

    if $\epsilon_t > \frac{\gamma}{\gamma+1}$ **then**
      return $\alpha_t = 0$
    end if
    Set a weak learner weight α_t.

$$\alpha_t = \frac{1}{2}[\log(\frac{1-\epsilon_t}{\epsilon_t}) + log(\gamma)] \tag{3.8}$$

    Update the data distribution weights

$$w_t \Leftarrow w_t . e^{(\alpha_t \mathcal{I}[y_i \neq h_t(x_i)])}$$

    Normalize w_t to form probability distribution
  end for
end for
Combine weak learners h_t into a strong learner $H(x)$

$$H(x) = \arg\max_{y} \sum_{t=1}^{T} \alpha_t h_t(x)$$

Algorithm 2 ISB Weight initialization

for $j = 1, \ldots, K$: **do**
  if $(p \neq 1)$ **then**
    Find the maximum and minimum weights in class j and calculate the mean.
    $I = \frac{max(w_T^j(i)) + min(w_T^j(i))}{2}$
    Initialize new samples x^j with I.
  else
    Initialize distribution weights $w_1^j(i) = 1/|\mathcal{D}_1|$
  end if
end for

tion of the data set. If this is combined with ensemble hypothesis of previous evaluation as in Learn++, the zero-weighted hypothesis problem becomes more severe. On the other hand, assigning minimum weights results in driving the weak classifier's focus to the old data and loss of new valuable information. During evaluation, the misclassified new samples will cause an increased weighted error and again an increased number of zero-weighted hypotheses. In both cases, if the algorithm is allowed to continue, it might need many iterations to be able to recover from this problem by successively updating the weights.

Initializing new samples with a weight of $\frac{1}{|\mathcal{D}_p|}$, where $|\mathcal{D}_p|$ is the size of the new batch, following AdaBoost and SAMME, results in a uniform distribution weights. Different classes have different distributions and degrees of hardness. While some classes might have samples with high weights, some other classes might be linearly separable. Assigning equal weights to samples of all classes is unjustified since these weights might be extremely high or low for some classes. It also might result in a biased selection process.

Motivated by these problems and our class-based weighted selection strategy, we developed a weight initialization method as presented in Algorithm 2. In the first iteration the sample weights are initialized according to the size of the first batch $w_1(i) = 1/|\mathcal{D}_1|$. When a new batch of data is available, the average weights in each class are used as the initial weights to the new examples belonging to their corresponding classes. Hence, the initialized weights could be different from class to class. As a result, for an individual class, the probability of selecting new examples in the next training round almost doubles that of the least weighted examples and is about the half of those with highest weights. The process allows a smooth transition by integrating new examples with the key examples in the existing data set. Using medium weight value to initialize the weights for the new examples avoids sudden increase of the weighted error.

4 Experiments and Discussion

Three data sets were prepared for our experiments; synthetic data set, Capsule Endoscopy videos, and UCI image segmentation data set. All experiments were conducted using two-fold cross validation and the average error rates were calculated. Other metrics used to evaluate the experiments are training efficiency, average effectiveness of new batch training E_n, and average effectiveness of training all iterations E_v. E_n: average number of non-zero weighted hypothesis within five iterations after the introduction of new batches of data. This measure indicates the degree of integrating new samples smoothly. E_v: average overall number of effective non-zero weighted hypothesis after training all

iterations.

All experiments used Decision Trees employing early pruning since boosting with decision trees is recognized as "the best off-the-shelf classifier" [2]. Incremental SampleBoost experiments employed several down sampling sizes per class s and different values for the error parameter γ. Results are presented for different γ that we randomly selected. For γ, we observed that lower values result in a deteriorated performance that is still better than other methods on average. Based on our preliminary experiments, we suggest using $\gamma \geq 0.4K$. We define a specific number of training iterations and keep count of all zero-weighted hypotheses. For the comparison, Learn++ is restricted to have at least one sample per class after the selection process to avoid loss of any classes during training.

A comparison of the average error rates with different initialization methods on the synthetic data set using our ISB method is provided in Figure 1. The curves show the average error rates when assigning the new samples minimum weights, maximum weights (as in Learn++), fixed weights (as in AdaBoost and SAMME), and our class-based weights denoted with Small W, Large W, Avg W, and C-B W, respectively. The comparison shows that for most of the classes our class-based weight initialization scheme achieves the lowest error rates. All other methods show very close performance and exchange the second best performance along different classes. This indicates that out strategy is the most effective in integrating new samples.

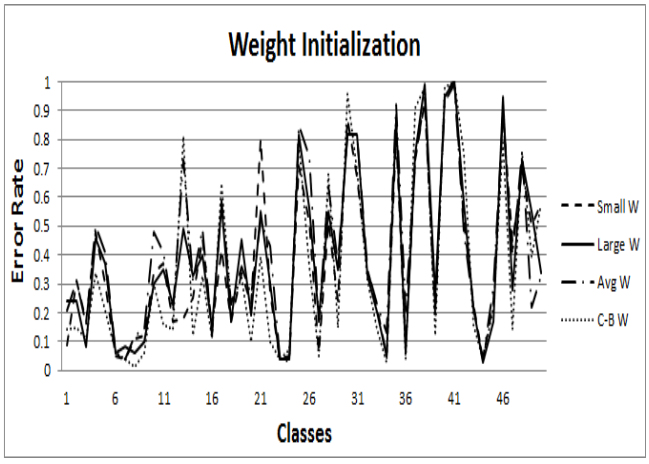

Figure 1: Weight Initialization.

The first set of experiments trained with a two dimensional (2D) Gaussian synthetic data set that was created using a Gaussian function and includes 50 different classes. The mean of each class was generated randomly and all classes have a unit covariance. Each class have a sample size of 100 to form a total size of 5000 samples. The data set was trained in balanced and imbalanced scenarios. Results of balanced training is shown in Figure 2 and Table 1. (V) in all tables denotes a variable sample size per class as new batches are introduced. Five batches of data were trained. Every 20 iterations a new batch was introduced for a total number of 100 training iterations. All batches have equal size including 10 samples per class for the 50 classes. Figure 2 compares the average error rates for individual classes using ISB, Learn++, Incremental AdaBoost.M1, and Incremental SAMME. ISB used an fixed under sampling size of $s = 8$ samples per class during all iterations with $\gamma = 30$. The results show that for most of the classes ISB achieves lower error rates. I-AdaBoost.M1 has the highest error rates. Learn++ and I-SAMME have very close performance.

Table 1 compares the performance of ISB using different class sizes and γ with Learn++ (L++), Incremental AdaBoost.M1 (I-A), Incremental SAMME (I-S), Full batch AdaBoost.M1 (FP-A), and Full batch SAMME (FP-S). The average error rates (AER) show that ISB using $s = 8$, with $\gamma_1 = 30$ and $\gamma_2 = 49$ achieves the lowest error rates. ISB, $s = 5$, using $\gamma_1 = 30$ has a slight increase in the error rate and I-AdaBoost.M1 has the highest error rate. E_n and E_v shows that ISB with different class sizes and γ have the highest effectiveness in integrating new samples. Learn++ comes second while I-AdaBoost.M1, Full batch AdaBoost, and SAMME show the least performance. Efficiency of all methods for the synthetic data sets is very close except for Learn++. This is explained by the ensemble evaluation schemes that are employed during every training round which results in a significantly lower efficiency.

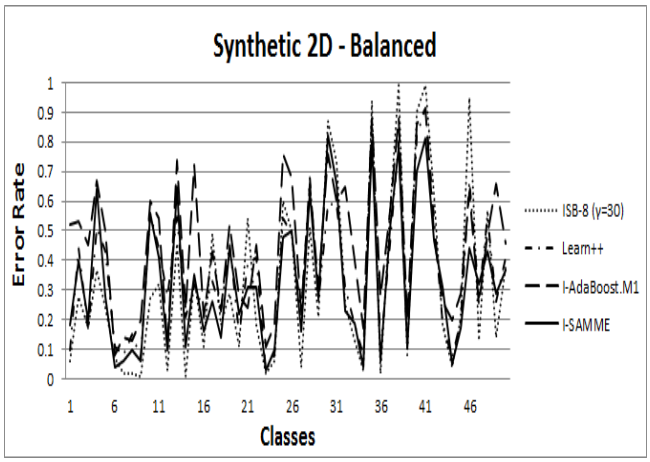

Figure 2: The average error rate for the synthetic data set.

For the imbalanced experiments, 25 classes were randomly selected to form the minority classes and

then switched with the other 25 classes so that every class is trained once as minority and once as majority. Five batches of data were used and each batch has two samples per minority class and 10 samples per majority class with a total size of 300 samples per batch. This forms an imbalance ratio of 1:5. ISB fixes s to the size of the least minority class during all iterations. Figure 3 shows the average minority error rates for all classes. Clearly ISB outperforms other methods along different classes. On the contrary, Learn++ suffers the most from imbalanced scenarios. This can be attributed to its selection scheme which introduces additional imbalance and bias towards some classes.

Table 1 shows that AER of ISB with $\gamma_1 = 30$ and $\gamma_2 = 49$ have the least minority error rate with error reduction of 46% compared to Learn++. While other methods show improvement for the majority classes, the overall AER still shows the superiority of ISB. E_n and E_v show similar performance as in the balanced training putting ISB in the lead of employing the most effective integration. Efficiency results also shows the low efficiency of Learn++ compared to other methods.

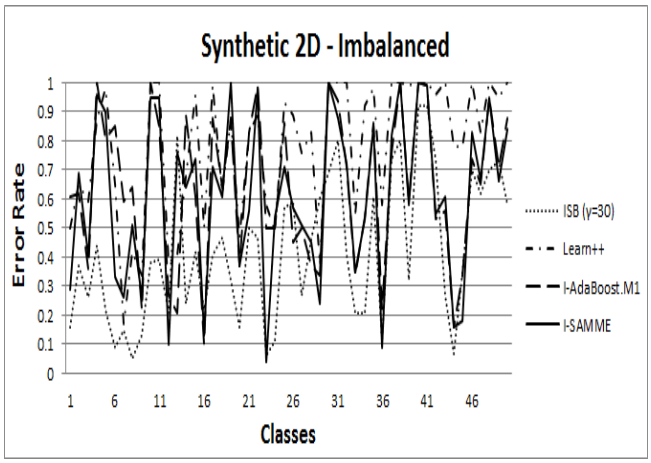

Figure 3: The minority error rate for the synthetic data set.

The second data set is Capsule endoscopy CE videos. Thousands of images are generated daily from capsule endoscopy patients. These images require a lot of human effort to detect abnormality like bleeding and tumors. With new examples generated, training a single model with all the data might be infeasible and therefore, incremental learning plays a very important role for improved classification.

CE videos were collected from eight different patients. The number of frames of each video ranges between 40,000 to 60,000. The CE experiments have two stages. The first classifies the frames into their respective organs along the digestive track. The process classifies the frames to stomach, small intestine, or large intestine in a balanced training. Each of these classes has 1200 images randomly selected to form a total sample size of 3600. All images were processed using Principal Component Analysis for dimensionality reduction. Three batches of data were trained for the experiments and each has 200 images per class. Each batch was trained for 30 iterations for a total of 90 iterations. Figure 4 shows that ISB ($s = 50$ and $\gamma = 2$) outperforms all other methods for all classes. Choices for γ here is 1 or 2. Learn++ has the second best performance for the small intestine and large intestine classes while incremental AdaBoost and SAMME have lower performance. The increase in the error rates of all methods except ISB for the large intestine class can be attributed to the presence of stools at this stage of the digestive tract. ISB appears to be robust to occlusions.

Table 1: Synthetic data set results. Number of training samples per class (S/C), average error rate (AER.%), minority error rate (Min%), majority error rate (Maj%), effective new batch training (E_n), overall effective training iterations (E_v), and efficiency in seconds (EFF.) for balanced and imbalanced (Imb) training.

Balanced	ISBγ_1	ISBγ_2	ISBγ_1	L++	I-A	I-S	FP-A	FP-S
S/C	8	8	5	V	V	V	50	50
AER%	32.4	32.5	34.1	35.8	44.6	33.4	34.6	34.6
E_n	3.5	3.9	3	3.4	0.5	2.5	0.6	0.3
E_v	66.5	65.5	76	40.5	3	12.5	3	1.5
EFF.	47.7	48.9	40.7	339	48.2	46.8	64.4	64.3

Imb 1:5	ISBγ_1	ISBγ_2	L++	I-A	I-S	FP-A	FP-S
S/C	V	V	V	V	V	10:50	10:50
AER%	39.8	41.0	51.3	46.9	42.1	42.8	42.8
Min%	42.0	42.0	77.9	64.4	60.8	60.5	60.5
Maj%	37.6	40.1	24.8	29.5	23.5	25.1	25.1
E_n	3.55	4.05	3.1	1.05	2.25	0.35	0.4
E_v	66.2	66.7	34	5.25	11.2	1.75	2.0
EFF.	42.5	42.3	224	30.0	30.1	42.5	42.5

Table 2 shows that ISB with a fixed $s = 50$ and $s = 150$ has a significant lower error rate compared to all other methods. The error reduction reaches 32%. E_n and E_v show a superior performance of ISB and specifically for $s = 50$ where all the 90 training iterations were effective. On the other hand, all other methods show a significantly lower performance in training iterations as well as integrating new samples. The results of CE classification also show that ISB has better efficiency compared to all other methods. Learn++ achieves the second best efficiency. The reason for this is that training weak classifiers with all samples in CE takes significantly longer time than the evaluation process done during each training iteration

in Learn++.

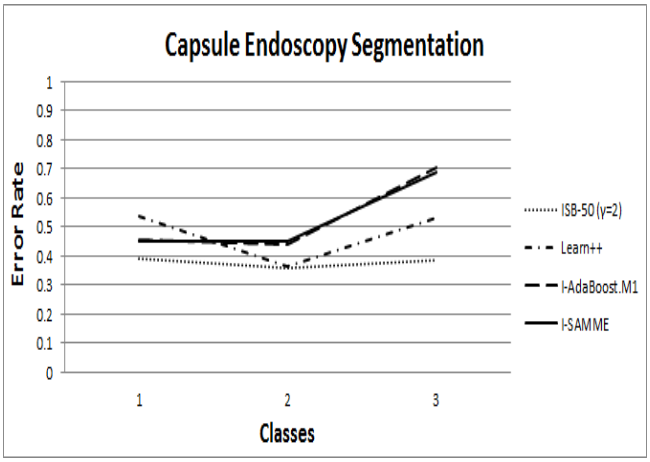

Figure 4: The average error rate for the CE data set.

Table 2: Capsule Endoscopy CE results.

Balanced	ISBγ_1	ISBγ_1	L++	I-A	I-S	FP-A	FP-S
S/C	50	150	V	V	V	600	600
AER%	38.0	39.3	47.9	53.3	52.8	56.1	56.1
E_n	5	5	2.66	3	2.1	0.6	1.5
E_v	90	87	21.5	9.5	6.5	2	5
EFF.	25.2	93.5	199	619	629	1271	1203

Imb 1:83	ISBγ_2	L++	I-A	FP-A
S/C	V	V	V	25:2075
AER%	2	50	50	49
Min%	4	100	100	98
Maj%	0	0	0	0
E_n	5	1.2	4.3	5
E_v	100	5.5	81.5	100
EFF.	12.5	66.3	66.4	105

The second stage of CE experiments divides the frames into normal and abnormal images in imbalanced training due to unavailability of abnormal images. Training of the normal class has 2075 images while that of the abnormal class has 25 images forming an imbalance ratio of 1:83. Along the digestive track, patients have massive number of normal images, and very few numbers of images containing diseased regions. We formed five batches of data and each contains five samples for the minority class and 415 for the majority. Each batch was trained for 20 iterations for a total number of 100 iterations.

Table 2 reveals a massive error reduction using ISB for the minority classes compared to all other methods. The error rate reduction is up to 96%. The majority class performance achieves 0% error rate for all methods. AER here is exceptionally calculated as the average of the minority and majority error rates. This was done since we only had 25 samples to test the minority class and 2075 to test the majority. Incremental and full batch AdaBoost.M1 and SAMME convert to the incremental and full batch binary AdaBoost, (I-A) and (FP-A) in the table. ISB has the maximum E_n and E_v while Learn++ has the lowest. It's also noticed that a high E_n and E_v was observed for I-A and FP-A due to the close to zero evaluation error during each training iteration in this binary classification.

The last dataset presented is UCI Image Segmentation. The data set consists of seven classes each having 330 samples to form a total sample size of 2310. A 2-fold cross validation was employed. Experiments were conducted using balanced and imbalanced training. In balanced training, five batches of samples were trained with 33 samples per class each. Each new batch was introduced after 20 training iterations for a total of 100 iterations. In these experiments, in addition to applying different s and γ, we also updated the value for s when a new batch is introduced and compared it with a fixed s. We start the variable s with 20 samples and add 20 additional samples when a new batch is introduced. This forms sample size of 20, 40, 60, 80, and 100 per class when each of the 5 batches is introduced. ISB with variable s is shown in Figure 5 and compared to other methods. The figure again shows the improvement obtained using ISB. The proposed method has the best performance for 5 of the 7 classes and the second best for 2 of them. Learn++ ranges between the best and worst performance across different classes. Incremental AdaBoost.M1 and SAMME show very close performance.

Table 3 compares the results of ISB, using variable (V) s and fixed s with $\gamma_1 = 3$ and $\gamma_2 = 6$, with all other methods. ISB with variable s outperforms the other methods in terms of AER. ISB with a fixed $s = 20$ comes second. ISB achieves up to 53% error reduction compared to the other methods. Although Learn++ achieves the highest E_n, ISB has a very close E_n and the highest E_v of all methods. Learn++ achieves the lowest efficiency. The full batch AdaBoost.M1 and SAMME are noticed to have very similar performance across different experiments. The reason can be seen in our weighted error analysis due to repetition of the same misclassified samples which results in increased weight error during most of the iterations.

In imbalanced training three classes were randomly selected to form the minority classes and then switched with other four classes so that all classes would be used as the minority and as the majority. Data was divided into five batches, each containing 11 samples for minority class and 33 samples for majority classes forming an imbalance ratio of 1:3. New data batches were introduced every 20 iterations for a total of 100

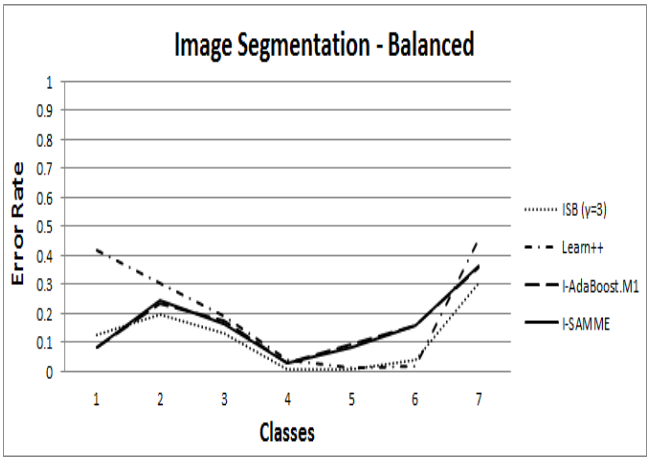

Figure 5: The average error rate for the image segmentation data set.

Table 3: UCI Image Segmentation results.

Balanced	ISBγ_1	ISBγ_1	ISBγ_2	L++	I-A	I-S	FP-A	FP-S
S/C	20	V	V	V	V	V	165	165
AER%	13.5	11.6	9.9	20.7	16.2	16.1	21.4	21.3
E_n	2.9	3.1	3.1	3.8	2.6	2.5	0.4	0.2
E_v	65.5	53.5	56.5	41	13.5	12.5	2	1
EFF.	7.02	11.9	12	96.7	9.3	9.04	13.6	13.5

Imb 1:3	ISBγ_1	ISBγ_2	L++	I-A	I-S	FP-A	FP-S
S/C	V	V	V	V	V	55:165	55:165
AER%	15.6	15.9	40.8	29.7	29.9	34.5	34.5
Min%	17.7	18.8	60.2	43.2	43.8	33.4	33.4
Maj%	13.5	13.1	21.5	16.1	16.0	35.6	35.6
E_n	4.1	4.35	3.55	2.55	2.35	0.4	0.45
E_v	73	73.7	40.5	14.2	11.7	2	2.5
EFF.	8.6	8.52	77.5	7.47	7.47	8.48	8.47

iterations. Figure 6 shows that ISB using $\gamma = 3$ outperforms all other methods in reducing the minority average error rates. Learn++ suffers most in these imbalanced scenarios.

In Table 3, ISB with different γ evidently achieves the lowest error rate for both the majority and minority classes. ISB also achieves the highest E_n and E_v showing its effectiveness in integrating new samples as well as the overall training process. Same efficiency trend was observed as in the balanced training results.

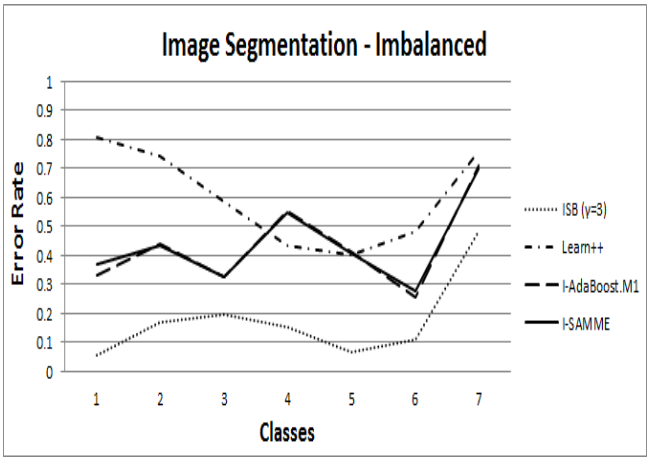

Figure 6: The minority error rate for the image segmentation data set.

5 Conclusion

Ensemble classification method has great potential to learn incrementally from new data. New batches of data can be introduced after certain number of training iterations and an overall hypothesis is created to classify unseen data. However, traditional ensemble classification methods such as AdaBoost and its multi-class extensions AdaBoost.M1 and SAMME suffer from repeatedly misclassified examples which increase the weighted error and cause termination of the algorithm. Also expensive computational resources are required to train both old and new samples combined. Moreover, the error condition suggested by some of these methods is too strict and contributes further to the termination problem. The weight initialization process poses another problem as it plays a very important rule in the algorithm ability of integrating new data.

Newer algorithms created to allow incremental learning using ensemble classification such as Learn++ suffers in generalizing well in multi-class classification using its down sampling mechanism. The method performance deteriorates significantly in imbalanced classification scenarios with its strict error condition. Although Learn++ shows improvement compared to some other methods, the improvement is inconsistent and relies on forgetting old data while keeping ensemble hypothesis and assigning maximum weights to new samples.

In this paper, we introduce Incremental SampleBoost, an ensemble learning method that efficiently learns from new data. The method employs a class-based down sampling mechanism depending on the weight distribution. The mechanism treats different classes independently and hence avoids introducing any bias towards any of the classes. Moreover, it balances classes in imbalanced scenarios. The down sampling strategy combined with an a variable error parameter introduces the destabilization required for the weak learner to generate different decision boundaries and accordingly avoid repetition of misclassified samples. Unlike other methods, this also allows integrating the most stable weak classifiers within the proposed framework.

The new method also introduces a novel weight initialization scheme that effectively integrates new samples.

Experimental results show the superiority of ISB to all other methods in both accuracy and efficiency for multi-class classification. The method also achieves a significant improvement in imbalanced scenarios. Additionally, ISB achieves the most effective training compared to other methods. ISB also shows a superior performance in the ability to smoothly integrate new data without increasing the weighted error and further wasting training iterations.

References

[1] M. Abouelenien and X. Yuan, *Sampleboost: Improving boosting performance by destabilizing weak-learners based on weighted error analysis*, in 21st International Conference on Pattern Recognition, ICPR 2012, November 2012.

[2] L. Breiman, *Arcing classifiers*, The Annals of Statistics, 26 (1998), pp. 801–824.

[3] C. Domeniconi and D. Gunopulos, *Incremental support vector machine construction*, in IEEE International Conference on Data Mining, 2001, pp. 589 –592.

[4] Y. Freund and R. E. Schapire, *A decision-theoretic generalization of on-line learning and an application to boosting*, Journal of Computer and System Sciences, 55 (1997).

[5] B. Giritharan, S. Panchakarla, and X. Yuan, *Segmentation of CE videos by finding convex skin*, in 2010 IEEE International Conference on Bioinformatics and Biomedicine Workshops (BIBMW), Dec. 2010, pp. 158 –163.

[6] F. H. Hamker, *Life-long learning cell structurescontinuously learning without catastrophic interference*, Neural Networks, 14 (2001), pp. 551 – 573.

[7] H. S. Mohammed, J. Leander, M. Marbach, and R. Polikar, *Can AdaBoost.M1 learn incrementally? a comparison to learn++ under different combination rules*, in International Conference on Artificial Neural Networks, ICANN 2006, vol. 4131, Springer, 2006, pp. 254–263.

[8] 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, 20 (2009), pp. 152–168.

[9] S. Ozawa, S. Pang, and N. Kasabov, *Incremental learning of chunk data for online pattern classification systems*, IEEE Transactions on Neural Networks, 19 (2008), pp. 1061–1074.

[10] R. Polikar, L. Upda, S. S. Upda, and V. Honavar, *Learn++: an incremental learning algorithm for supervised neural networks*, IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, 31 (2001), pp. 497 –508.

[11] A. Shilton, M. Palaniswami, D. Ralph, and A. C. Tsoi, *Incremental training of support vector machines*, IEEE Transactions on Neural Networks, 16 (2005), pp. 114 –131.

[12] N. A. Syed, S. Huan, L. Kah, and K. Sung, *Incremental learning with support vector machines*, in Workshop on Support Vector Machines at the International Joint Conference on Articial Intelligence (IJCAI-99), 1999.

[13] P. E. Utgoff, N. C. Berkman, and J. A. Clouse, *Decision tree induction based on efficient tree restructuring*, Machine Learning, 29 (1997), pp. 5–44. 10.1023/A:1007413323501.

[14] Z. Zhou and Z. Chen, *Hybrid decision tree*, Knowledge based system, 15 (2002), pp. 515–528.

[15] J. Zhu, H. Zou, S. Rosset, and T. Hastie, *Multi-class adaboost*, Statistics and Interface, 2 (2009), pp. 349–360.