A New Incremental Learning Algorithm with Probabilistic Weights Using Extended Data Expression

Kwangmo Yang¹, Anastasiya Kolesnikova², and Won Don Lee^{3*}, *Member, KIICE*

¹The Attached Institute of ETRI, Daejeon 305-600, Korea

²TV Software Platform Research Department, LG Electronics, Seoul, Korea

³Department of Computer Science, Chungnam National University, Daejeon 305-764, Korea

Abstract

New incremental learning algorithm using extended data expression, based on probabilistic compounding, is presented in this paper. Incremental learning algorithm generates an ensemble of weak classifiers and compounds these classifiers to a strong classifier, using a weighted majority voting, to improve classification performance. We introduce new probabilistic weighted majority voting founded on extended data expression. In this case class distribution of the output is used to compound classifiers. UChoo, a decision tree classifier for extended data expression, is used as a base classifier, as it allows obtaining extended output expression that defines class distribution of the output. Extended data expression and UChoo classifier are powerful techniques in classification and rule refinement problem. In this paper extended data expression is applied to obtain probabilistic results with probabilistic majority voting. To show performance advantages, new algorithm is compared with Learn++, an incremental ensemble-based algorithm.

Index Terms: C4.5, Ensemble-based algorithm, Incremental learning, Learn++

I. INTRODUCTION

In many practical applications, where data become available in small series from time to time, the problem, known as “catastrophic forgetting”, requires data to be learned incrementally without losing classification performance on old data. Traditional classification algorithms do not allow incremental learning. When new data become available, these algorithms are reinitialized and retrained using a combination of old and new data. The problem becomes unsolvable, if old data are not available any more. Under such condition, it is necessary to update an existing classifier in an incremental way to accommodate new data without forgetting previously acquired knowledge.

There are several approaches for incremental learning. One of them is using recently developed ensemble-based algorithms that make creative use of bootstrap based ideas. First, this approach involves generating an ensemble of classifiers for each dataset that becomes available. The result is an ensemble of ensembles of classifiers. And then classifiers are compounded to a final classifier.

One of the main characteristics of ensemble is diversity among classifiers. It is obvious because combining same classifiers doesn't lead performance improvement. Diversity among classifiers can be achieved by adapting the classifiers to the training data distribution to create different training data subsets. This procedure guarantees that classifiers focus on more informative instances.

Received 19 March 2013, Revised 21 June 2013, Accepted 12 July 2013

*Corresponding Author Won Don Lee (E-mail: wlee@cnu.ac.kr, Tel: +82-42-821-5441)

Department of Computer Science, Chungnam National University, 99 Dachak-ro, Yuseong-gu, Daejeon 305-764, Korea.

Open Access <http://dx.doi.org/10.6109/jicce.2013.11.4.258>

print ISSN: 2234-8255 online ISSN: 2234-8883

© This is an Open Access article distributed under the terms of the Creative Commons Attribution Non-Commercial License (<http://creativecommons.org/licenses/by-nc/3.0/>) which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.

Copyright © The Korea Institute of Information and Communication Engineering

There are several algorithms to compound classifiers. In the simple majority voting, a commonly used combination rule, each classifier votes for the class it predicts, and the class receiving the largest number of votes is the ensemble decision. In the weighted majority voting, each classifier has a voting weight inversely proportional to its classification error. The winner is a class with the largest total vote. Also, algebraic combination of the class values can be used, so that the winner is a class receiving the highest combined vote [1]. All these algorithms consider just one class as a result of classifier, while the classifier can give class distribution as a result. For example, the leaf of decision tree, in case of weak classifier, can contain events from all classes. Losses of information are unavoidable, when the winner class is chosen. An alternative approach is to consider the class distribution.

A new probabilistic majority voting, based on probabilistic output expression, is presented in this paper, in the incremental learning.

Effectiveness of the extended data expression was shown previously [2, 3]. In this paper, a new application of extended data expression in combining the outputs of weak classifiers is shown to be effective.

The rest of the paper is organized as follows. First, an extended data expression is described. Second, UChoo, a decision tree classifier using the entropy measures as same as the ones used in C4.5, but with the extended data expression, is described. Next, a new incremental learning algorithm, founded on probabilistic majority voting, is proposed. Finally, experimental results are provided to show the performance of the new algorithm in comparison with Learn++ incremental learning algorithm.

II. RELATED WORK

Learn++ is an incremental learning algorithm that was introduced by Polikar et al. [4-6]. It is based on the AdaBoost, which uses multiple classifiers to allow the system to learn incrementally. The algorithm works on the concept of using many classifiers that are weak learners to give a good overall classification. The weak learners are trained on a separate subset of the training data and then the classifiers are compounded using a weighted majority voting. The weights for the weighted majority voting are calculated using the performance of the classifiers on the entire training data set.

III. EXTENDED DATA EXPRESSION

First, consider a simple example of data in common expression. Table 1 consists of training data, where each datum is described by three attributes, two of which are

discrete, while the 3rd is a continuous attribute, and a class. The “Outlook” attribute has three outcomes of ‘sunny’, ‘overcast’ and ‘rain’. “Windy?” has two outcomes of ‘True’ and ‘False’. Both “Outlook” and “Windy?” are discrete attributes and “Temperature” is a continuous attribute whose values range from 60 to 80. Finally, class has two outcomes of ‘Play’ and ‘Don't Play’.

Table 2 shows the modified expression [2] of the training data set of Table 1. There are two main features of extended expression. First, each attribute has distribution over the possible values, i.e. each attribute entry is to be filled with a probabilistic value ranging from 0 to 1. Second, each event has a weight value, which shows how much importance the event has. In this example, all the data are assumed to have the equal weight of one.

Data can be described by an expert in the field. For example, an expert says that it is ‘sunny’ and the temperature is around 70 and the probability to play is 2/3 regardless of whether windy or not.

In addition, the expert can say that the data or ‘rule’ he describes has the importance of having the weight of 30 compared to the individual datum in the data set. UChoo can then express the case as the event #1 shown on the first row of the Table 3.

Since each event has a weight, an instance has been redefined as the event with a weight value of 1 in Table 3. In other words, each attribute can have distribution over the possible class values in an instance. And each row of Table 3 has a weight value, which shows how much importance the event has. For example, if each observed instance is assumed to have the importance weight of 1, the expert can now be able to describe his idea or rule’s importance with the weight value so that it can be compared with an observed instance. The event at the first row of the above Table 3 has the weight value of 30. In other words, the event has the importance equivalent to that of 30 instances.

An event is a collection of instances with equal attribute value distribution and class value distribution.

An instance can be regarded as an instance having the weight of 1. Therefore, the number of events may not be equal to that of all the instances. For example, in Table 3, the number of all the instances is 35 while the number of the events is 6. The number of all the instances in a data set T is denoted as $|T|$ in this paper while the number of the events is denoted as p .

Table 1. Training data set

Outlook	Temp (°F)	Windy?	Class
Sunny	70	False	Don't Play
Sunny	60	True	Play
Overcast	80	False	Play
Rain	60	True	Don't Play
Rain	70	False	Play
Rain	80	True	Don't Play

Table 2. Extended data expression

Event#	Weight (i)	Outlook			Temp (°F)			Windy?		Class	
		Sunny	Overcast	Rain	60	70	80	True	False	Play	Don't Play
1	1	1	0	0	0	1	0	0	1	0	1
2	1	1	0	0	1	0	0	1	0	1	0
3	1	0	1	0	0	0	1	0	1	1	0
4	1	0	0	1	1	0	0	1	0	0	1
5	1	0	0	1	0	1	0	0	1	1	0
6	1	0	0	1	0	0	1	1	0	0	1

Table 3. Extended data expression with the expert knowledge

Event#	Weight (i)	Outlook			Temp (°F)			Windy?		Class	
		Sunny	Overcast	Rain	60	70	80	True	False	Play	Don't Play
1	30	1	0	0	0	1	0	1/2	1/2	2/3	1/3
2	1	1	0	0	1	0	0	1	0	1	0
3	1	0	1	0	0	0	1	0	1	1	0
4	1	0	0	1	1	0	0	1	0	0	1
5	1	0	0	1	0	1	0	0	1	1	0
6	1	0	0	1	0	0	1	1	0	0	1

UChoo algorithm utilized this extended data expression to make the classifiers from incomplete, missing or varying dimensional data as in [2, 3].

IV. UCHOO ALGORITHM

UChoo algorithm uses the measure entropy similar to that of C4.5 [7], but it not only can generate classifiers from the observed data set as in C4.5, but also from the combination of the observed data with the rules or data from the expert described in the form of the extended data set expression.

The entropy measure equation of C4.5 is as follows:

$$Gain_ratio(A) = Gain(A)/Split_info(A), \quad (1)$$

where *Gain_ratio* is used as the measure of how much an attribute, *A*, can differentiate the classes. *Gain(A)* is the mutual information between the class and the attribute *A*, and *Split_info(A)* is the normalization constant so that the number of the outcomes of attribute *A* does not affect the information measure.

$$Gain(A) = info(T) - info_A(T). \quad (2)$$

$$info(T) = -\sum_{i=1}^k (freq(C_i, T)/|T|) \cdot \log_2(freq(C_i, T)/|T|). \quad (3)$$

$$info_A(T) = \sum_{j=1}^k |T_{A_j}|/|T| \cdot info(T_{A_j}). \quad (4)$$

$$split_info(A) = -\sum_{i=1}^k |T_{A_j}|/|T| \cdot \log_2(|T_{A_j}|/|T|), \quad (5)$$

where *A* is an attribute, and *k* is the number of values of a class, and *n* is the number of outcomes of the attribute.

New equations for extended data expression are defined as follows.

Class membership weight $C_i(m)$ states how much the *m*th event belongs to the class C_i and the following condition must be satisfied:

$$\sum_{i=1}^k C_i(m) = 1. \quad (6)$$

Outcome membership weight $O_{A_j}(m)$ states how much the outcome value *j* in the attribute *A* can happen in the *m*th event and the following condition must be satisfied:

$$\sum_{j=1}^k O_{A_j}(m) = 1. \quad (7)$$

T_{A_j} is the subset of *T* that has the outcome value *j* in the attribute *A*. For example, as the attribute *A* has three values, the set *T* is divided into three subsets. The $freq(C_i, T)$ is the number of instances in the set *T*, which have the class value of C_i , as follows:

$$freq(C_i, T) = \sum_{m=1}^{p(T)} Weight(m, T) \cdot C_i(m), \quad (8)$$

where $Weight(m, T)$ is the weight value of the *m*th event in the set *T*. $|T|$ is the number of instances in the set *T*. An instance means the event whose weight is 1. Therefore, if an event has the weight *W*, it means that there are *W* number of instances with equal distribution values of attributes and class. $|T_{A_j}|$ is the number of instances in the set T_{A_j} , as follows:

$$|T_{A_j}| = \sum_{m=1}^{p(T_{A_j})} Weight(m, T_{A_j}) \cdot O_{A_j}(m). \quad (9)$$

In this case, multiplying $O_{A_j}(m)$, the outcome possibility value of each event in T_{A_j} , by $Weight(m, T_{A_j})$ for each *m* and adding the results of all the multiplications, we get $|T_{A_j}|$. Therefore, the best attribute for splitting has the biggest *Gain_ratio* in the node of the recursion tree by using the

entropy equations with these newly defined values.

V. INCREMENTAL LEARNING ALGORITHM BASED ON NEW PROBABILISTIC MAJORITY VOTING

A. Algorithm

Ensemble-based algorithms create collection of weak classifiers and compound them to strong classifier. Such algorithms use weighted majority voting, based on the performance of the classifiers. In this case, a weak classifier gives out a hypothesis and decides one class for each training event.

Consider a situation for a decision tree: a leaf contains 100 events and 45 of them belong to class 1, the other 55 events belong to class 2. So, hypothesis assigns this leaf to class 2 as class 2 has more popularity in the leaf.

Probabilistic expression is more natural. For the previous example, weak classifier gives class distribution as hypothesis: 0.45 for class 1, 0.55 for class 2. This class distribution can be used to compound hypotheses.

Consider the example in Fig. 1. Here, n is the number of classifiers, k is the number of classes, m is the number of databases. $\{w_i^1, \dots, w_i^n\}$ is class distribution of C_i^t , C^t is compound hypothesis of $C_1^t, \dots, C_n^t$ and the weights in the compound distribution can be calculated as follows using the error of the j th classifier e^j .

$$w_i = \frac{\sum_{j=1}^n \log((1-e^j)/e^j) w_i^j}{K}. \quad (10)$$

Probabilistic expression is easily realized by using UChoo classifier and extended data expression. Weights of events in extended data expression can be used to define class distribution as a result of classifier. UChoo is the classifier based on such data expression and the class distribution in a leaf can be expressed as follows:

$$w_i^{leaf} = \frac{\sum_{m \in T_{leaf}} \text{Weight}(m) \cdot C_i(m)}{K}, \quad (11)$$

Fig. 1. Ensemble of classifiers.

where T_{leaf} is the set of events in the leaf and K in Eqs. (10) and (11) is a normalized constant to make the sum of all the weights equals one, $\sum_{i=1}^k w_i = 1$.

Define here the training data $= \{(w_i, x_i, y_i)\}_{i=1}^n$, where $w_i \geq 0$ is the weight of instance, $x_i \in X$ is the i th instance in the extended attribute space X and $y_i \in Y$ is the class label of x_i in the extended class space Y . T is number of weak classifiers. Output of WeakLearn for the event x is hypothesis $h(x) = (h_1(x), \dots, h_k(x))$, where $h_i(x)$ states how much the event x belongs to the class C_i and $\sum_{i=1}^k h_i(x) = 1$.

UChoo classifier with pruning parameter is used as WeakLearn. Pruning parameter controls the weakness of a classifier. Pruning parameter controls the ratio of the majority class instances to the total instances in the leaf. Pruning 40% means, that each leaf consists of 60% of one class. Class distribution in each leaf is determined by the formula described above.

The block diagram of the incremental learning is in Fig. 2, and the algorithm is described in Fig. 3.

At first, for available database, the distribution is initialized as uniform, and all the events have equal probability to be chosen into training subset TR_t . At each iteration, weak classifier (UChoo) is trained on TR_t to obtain hypothesis $h^t = \{h^t(x_i) | x_i \in S\}$. Compounded hypothesis is calculated according to the normalized class distribution in a leaf given above.

Weights are changed by using performance of compounded classifier applying AdaBoost rule [8], so that the weights of misclassified events are increased relative to the correctly classified training events. In AdaBoost weights are changed according to performance of the hypothesis, which is result of current weak classifier. Here the performance of the compounded hypothesis is used to update weights, so the ability of resulting hypothesis is improved after each iteration. Also, resampling with replacement is used to obtain training subset TR_t . So, the events difficult to be trained can be chosen to the training set more than one time. It means an instance will have higher weight, when it is incorrectly classified, and UChoo classifier focuses on such an event. In the end, final hypothesis is obtained by compounding hypotheses for all the databases.

To show the power of proposed probabilistic result expression, new incremental algorithm, explained in this section, is compared with Learn++ with UChoo classifier used as weak classifier. Learn++ algorithm has the same plan, but uses deterministic weight majority voting.

B. Analysis

Since we use the same scheme as Learn++, error bound is defined in the same way. At first, consider a two-class problem, where there are only class 0 and class 1. In this

Fig. 2. Block diagram of incremental learning.

Do for each database DB_m , $m = \overline{1, M}$
 Initialize $w_i^1 = D_i^1 = \frac{1}{n}$
 for $t = \overline{1, T}$
 1. $D_t(i) = \frac{w_t(i)}{\sum_{j=1}^n w_t(j)}$
 2. Choose training subset TR_t according to D_t
 3. Train UchooLearn with TR_t
 and obtain hypothesis $h^t = \{h^t(x_i) | x_i \in DB_m\}$
 4. Compute error of h^t :

$$e^t = \sum_{i: \arg \max_j H_j^t(x_i) \neq \arg \max_j y^j(x_i)} D_i^t$$

 5. Compound hypothesis

$$H_i^t(x) = \frac{\sum_{j=1}^t \log((1 - e^t)/e^t) h_i^j(x)}{K}, x \in DB_m$$

 6. Compute error of H^t :

$$E^t = \sum_{i: \arg \max_j H_j^t(x_i) \neq \arg \max_j y^j(x_i)} D_i^t$$

 7. Update weights :

$$w_i^{t+1} = \begin{cases} w_i^t, & \text{if } (\arg \max_j H_j^t(x_i) \neq \arg \max_j y^j(x_i)); \\ \frac{E^t}{1 - E^t} \cdot w_i^t, & \text{otherwise} \end{cases}$$

 end for
 end Do

$$H_i^{final}(x) = \frac{\sum_{m=1}^M \sum_{j=1}^T \log((1 - E^j)/E^j) H_i^j(x)}{K}$$

$$H^{final}(x) = \arg \max_{C_i} H_i^{final}(x)$$

Fig. 3. The proposed algorithm.

case hypothesis $h_1^t(x)$ denotes probability of class 1 for the event x . Accordingly, probability of class 0 is $h_0^t(x) = 1 - h_1^t(x)$.

Error bound of compound hypothesis H^t obviously is AdaBoost error bound [8], that is,

$$E^t \leq 2^t \prod_{i=1}^t \sqrt{e^i(1 - e^i)}. \quad (12)$$

The final hypothesis is

$$\begin{aligned} H^{final}(x) &= \arg \max_{C_i} H_i^{final}(x) \\ &= \begin{cases} 1, & \text{if } H_1^{final} > H_0^{final} \\ 0, & \text{otherwise} \end{cases} \\ &= \begin{cases} 1, & \text{if } H_1^{final} > 1/2 \\ 0, & \text{otherwise} \end{cases} \\ &= \begin{cases} 1, & \text{if } \frac{\sum_{j=1}^T \log((1 - E^j)/E^j) H_1^j(x)}{\sum_{j=1}^T \log((1 - E^j)/E^j)} > 1/2 \\ 0, & \text{otherwise} \end{cases} \end{aligned} \quad (13)$$

It is the same as the one in [6], and therefore the error bound is $E \leq 2^T \prod_{t=1}^T \sqrt{E^t(1 - E^t)}$.

In case of more than 2 classes, multiclass problem can be reduced to two-class problem with same error bound as it is shown in [8].

Notice that we define upper-bound of error, which says that probabilistic majority is not worse than the deterministic one. In [9], authors show that summing the out-

comes of the real-valued neural networks and then selecting the best prediction performs better than selecting the best prediction of each network and then combining them with a majority rule. The experiments in the next section are carried out in the same way.

VI. EXPERIMENTS

Algorithms are tested on data sets collected in UCI Machine Repository [10]: Glass (214 events), Balance Scale (625 events), Vehicle (846 events) and Pendigits (10992 events). The data sets do not contain any missing value or unknown value. Detailed descriptions and performance on each data set are explained in the following.

As the goal of the experiments is to show the advantages of new probabilistic majority voting, the algorithm is compared with Learn++. Experiments are made under equal conditions for both algorithms: UChoo classifier is used as WeakLearner, the same number of hypotheses is generated and the training set and the testing set are the same for each experiment.

At first, each data set is randomly divided into two subsets. First (80%) is used to train the classifier and second (20%) to test the algorithms. To show the incremental effect, training data set is divided into 3 data bases and organized in the way that the previously seen data are not used in the next stages of learning. Exception is made for Glass data. Training data set in this case is not divided into 3 data bases, because the number of events is small.

For each database 12 weak hypotheses are generated. Weakness of hypotheses varies with pruning parameter between 0% and 40%. Number of weak hypotheses is set to 12. All the experiments show that the error just slightly varies depending on the number of hypotheses in the range from 10 to 20. More than 20 hypotheses make the system work very slowly.

A. Glass Data

Data set consists of 214 events. 172 events are used for training and the remaining events are used for testing. Each event consists of 10 attributes, with 6 classes.

Table 4 shows the performance of algorithms on test data using different pruning parameters. Performance of probabilistic incremental algorithm is quite better than that of Learn++ with Uchoo as the base classifier for all tests. Also note that the error is slight less for weak learners when pruning parameter varies from 25 to 40%.

It means strong classifiers tend to lead to overfitting and slight diversity among classifiers. Experiments show an advantage of probabilistic majority voting over weight majority voting.

Table 4. Error on glass data

Pruning (%)	New incremental algorithm error	Learn++ (UChoo) error
40	26.1905	31.6761
35	21.4286	31.9047
30	23.8095	32.3809
25	28.5714	32.3809
20	30.7143	33.3333
15	30.9524	30.9524
10	30.9524	34.7619
5	19.0476	31.9047
0	25.9524	31.9047

B. Vehicle Data

Data set consists of 846 events. Among them, 677 events are used for training and the remaining events are used for testing. Data event is described by 18 attributes from which one of 4 classes is determined.

Table 5 shows performance of the algorithms on test data using different pruning parameters. We can observe better performance of probabilistic incremental algorithm compared to that of Learn++, but errors are similar for all values of pruning parameters.

In Tables 6 and 7, incremental ability of the two algorithms is compared. Third column shows average error of weak classifiers learned on each data base. 4th, 5th, and 6th columns show the performance of the classifier at each incremental step on all databases.

Table 5. Error on vehicle data

Pruning (%)	New incremental algorithm error	Learn++(UChoo) error
40	27.7515	28.6390
35	25.4438	28.2840
30	23.6623	31.3624
25	22.4852	26.1538
20	21.8935	23.6686
15	20.7101	26.1538
10	18.3432	24.4970
5	22.4852	25.4438
0	22.4852	22.7218

Table 6. Error of probabilistic incremental learning on vehicle data (pruning 30%)

Dataset	Number of events	Average learner	1	2	3
1	226	50.84	0	6.19	11.94
2	226	47.75	-	16.81	16.37
3	225	46.55	-	-	18.22
Test	169	55.75	33.13	30.17	23.66

Table 7. Error of Learn++ on vehicle data (pruning 30%)

Dataset	Number of events	Average learner	1	2	3
1	226	48.15	0	0.44	6.63
2	226	45.64	-	24.77	20.79
3	225	46.18	-	-	24.44
Test	169	55.52	35.50	32.54	31.36

Table 8. Error on balance-scale data

Pruning (%)	New incremental algorithm error	Learn++(UChoo) error
40	14.7646	15.3647
35	14.0041	16.7265
30	14.4823	15.2045
25	15.5234	15.6004
20	16.0037	16.0043
15	15.4072	16.6026
10	16.0084	16.2447
5	16.4852	17.4438
0	16.9852	17.72188

Table 9. Error of probabilistic incremental learning on balance-scale data (pruning 30%)

Dataset	Number of events	Average learner	1	2	3
1	167	38.42	0	13.17	13.17
2	167	33.13	-	5.38	8.38
3	166	36.19	-	-	10.84
Test	125	45.08	25.6	17.6	14.48

Table 10. Error of Learn++ on balance-scale data (pruning 30%)

Dataset	Number of events	Average learner	1	2	3
1	167	38.77	0	11.37	14.97
2	167	27.29		10.17	10.17
3	166	33.98			10.24
Test	125	42.97	26.4	21.6	15.20

Performance is increasing incrementally from one data base to another. The new algorithm gives 10% of improvement while Learn++ gets 4% on testing data at the final incremental step.

C. Balance-Scale Data

Balance Scale Weight & Distance data set, obtained from UCI repository, is generated to model psychological experimental results. The dataset consists of 625 events and has 5 attributes; 500 events are used for training and 125 events for testing.

Table 8 compares errors of the new algorithm and Learn++ for pruning parameter in the range of [0,40]. A better performance of the new algorithm is observed. Both algorithms show the advantage of combining weak classifiers to make a better compound classifier. In Tables 9 and 10, 11% performance improvement is observed on testing data.

D. Pendigits Data

Data set consists of 10992 events. 8794 events are used for training and remaining 2198 events are used for testing. Data set consists of 16 attributes which describe handwritten numerical characters from 0 to 9.

Table 11 shows performance of algorithms on test data using different parameters. Error of new boosting algorithm is less than that of Learn++.

Tables 12 and 13 show a 3% performance improvement in incremental ability. Errors of compound classifiers are shown in Table 14. Errors are decreasing through iteration. This demonstrates that algorithm is learning incrementally.

Table 11. Error on pendigits data

Pruning (%)	New incremental algorithm error	Learn++(UChoo) error
40	5.6415	7.0110
35	5.3230	6.9342
30	2.6837	3.0979
25	3.1847	6.5987
20	3.6397	6.6424
15	2.8207	7.9618
10	2.1838	7.4158
5	2.1383	5.6870
0	3.7307	1.7288

Table 12. Error of probabilistic incremental learning on pendigits data (pruning 30%)

Dataset	Number of events	Average learner	1	2	3
1	2932	35.60	1.26	1.94	1.77
2	2931	39.79	-	1.60	1.53
3	2931	35.69	-	-	2.18
Test	2198	38.53	5.73	3.00	2.68

Table 13. Error of Learn++ on pendigits data (pruning 30%)

Dataset	Number of events	Average learner	1	2	3
1	2932	32.83	2.83	2.55	2.62
2	2931	36.41		2.52	2.66
3	2931	34.42			2.45
Test	2198	36.32	6.23	3.36	3.09

Table 14. Error of new boosting algorithm through iterations on pendigits data

Number of Iteration	error	Number of Iteration	error
1	18.6069	7	6.7334
2	13.0573	8	4.8681
3	6.86990	9	4.5041
4	5.50500	10	4.5951
5	8.32580	11	4.1401
6	7.18840	12	4.8681
		final	3.0937

Table 15. Error of Learn++ through iterations on pendigits data

Number of Iteration	error	Number of Iteration	error
1	18.6533	7	9.4177
2	19.2903	8	8.5532
3	11.9199	9	8.1893
4	9.96360	10	7.5523
5	16.1965	11	6.8244
6	8.46220	12	6.8699
		final	5.0500

Tables 14 and 15 show errors of both algorithms through iterations on the testing set of Pendigits data. The two algorithms have almost the same error on first iteration; final hypothesis of new algorithm has better performance.

All the results show advantage of probabilistic majority voting over weight majority voting.

E. Overall Performance

Figs. 4 through 7 compare the new proposed algorithm with Learn++ for different pruning parameters. The x coordinate of the figures is (100-pruning parameter (%)) in percentage. New algorithm shows better results in almost all the experiments for various pruning parameters. Also, in Tables 6, 9, and 12, the performance of the hypothesis is higher on the own training data set and performance on the test set is improving as the hypothesis trained on new data base is added. In the Pendigits data the improvement is only 3%, but for vehicle and balance data it is 10%.

In the Pendigits data case, however, the error rate is decreased from 6.23% to 3.09%, which is actually a significant performance improvement as the error rate is decreased more than 50%.

VII. CONCLUSION

This paper presents a new incremental algorithm based on probabilistic result expression. Such expression prevents

losses of information in weight majority voting. Results of experiments corroborate advantage of the probabilistic result expression over the deterministic expression.

Fig. 4. Error rate on glass data. **Fig. 5.** Error rate on vehicle data. **Fig. 6.** Error rate on balance-scale data.

Fig. 7. Error rate on pendigits data.

Extended data expression and UChoo classifier are powerful techniques in classification and rule refinement problem. In this paper extended data expression is applied to obtain probabilistic results with probabilistic majority voting.

Proposed algorithm shows good performance that confirms the usefulness of extended data expression and UChoo classifier in solving classification problems.

To be able to calculate the probabilistic majority voting, the number of instances in the leaf should be counted. Because UChoo divides the space recursively according to the entropy measure, the subspace in the leaf is captured naturally and the probability of a certain class is seen. For a classifier such as artificial neural network, it is not easy to define the subspace where the test data belongs, and hence the probabilistic majority voting is not done naturally. This fact signifies the advantage of the decision tree type classifiers as a basic building block of more complicated ensemble of classifiers

ACKNOWLEDGMENTS

This work was supported by research fund of Chungnam National University.

Kwangmo Yang

is born in Okcheon, Korea in 1973. He received B.S. and M.S. degrees in Computer Science from Chungnam National University, Daejeon, Korea in 1998 and 2000, respectively. He is currently a senior member of engineering staff at The Attached Institute of ETRI. His research interests include data mining, machine learning and computer network.

REFERENCES

- [1] R. P. W. Duin, "The combining classifier: to train or not to train?," in *Proceedings of the 16th International Conference on Pattern Recognition*, Quebec, Canada, pp. 765-770, 2002.
- [2] J. M. Kong, D. H. Seo, and W. D. Lee, "Rule refinement with extended data expression," in *Proceedings of the 6th International Conference on Machine Learning and Applications*, Cincinnati: OH, pp. 310-315, 2007.
- [3] D. H. Kim, D. H. Lee, and W. D. Lee, "Classifier using extended data expression," in *Proceedings of the IEEE Mountain Workshop on Adaptive and Learning Systems*, Logan: UT, pp. 154-159, 2006.
- [4] R. Polikar, "Bootstrap-inspired techniques in computational intelligence," *IEEE Signal Processing Magazine*, vol. 24, no. 4, pp. 59-72, 2007.
- [5] R. Polikar, L. Udpa, S. S. Udpa, and V. Honavar, "Learn++: an incremental learning algorithm for multilayer perceptron networks," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, Istanbul, Turkey, pp. 3414-3417, 2000.
- [6] 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, no. 4, pp. 497-508, 2001.
- [7] J. R. Quinlan, "Bagging, boosting, and C4.5," in *Proceedings of the 13th National Conference on Artificial Intelligence*, Portland: OR, pp. 725-730, 1996.
- [8] R. E. Schapire, "The boosting approach to machine learning: an overview," in *MSRI Workshop on Nonlinear Estimation and Classification*, Berkeley: CA, 2002.
- [9] Y. L. Murphey, Z. Chen, and L. Feldkamp, "Incremental neural learning using AdaBoost," in *Proceedings of the International Joint Conference on Neural Networks*, Honolulu: HI, pp. 2304-2308, 2002.
- [10] UCI machine learning repository [Internet]. Available: <http://archive.ics.uci.edu/ml/>.

Anastasiya Kolesnikova

was born in Berdsk, Novosibirsk region, Russia in 1985. She earned her B.S. from Novosibirsk State Technical University, Russia in 2006, M.S. from Chungnam National University, Korea in 2009, both in Computer Science. After graduation she started working on database security at Korean company, KSign and then in 2011 she got a position of research engineer at LG Electronics, LCD TV department, where she is working now. Her main research interest areas are: Database, Data analysis, Incremental learning.

Won Don Lee

is born in Pusan, Korea in 1956. He earned B.S. from Seoul National University, Korea in 1979, M.S. from University of Illinois at Urbana in 1982, both in Chemistry, and Ph.D. from University of Illinois at Urbana in Computer Science, in 1986. He was awarded the graduate fellowship while attending the University of Illinois at Urbana. After getting Ph.D., he became a professor at the Department of Computer Science Engineering, University of Texas at Arlington before coming to Chungnam National University to serve as a professor in 1987. His main research interest areas are: machine learning, multimedia, and financial engineering.