International Journal on Artificial Intelligence Tools
Vol. 22, No. 4 (2013) 1350025 (17 pages)
© World Scientific Publishing Company
DOI: 10.1142/S0218213013500255

A NEW ENSEMBLE LEARNING ALGORITHM USING REGIONAL CLASSIFIERS

BYUNGWOO LEE

*Samsung Electronics Co., Ltd., 416, Maetan 3-dong, Yeongtong-gu,
Suwon-si, Gyeonggi-do, 443-803, Korea
bw1212@samsung.com*

SUNGHA CHOI

*LG Electronics, Inc., 20, Youido-dong, Yongdungpo-gu, Seoul 150-721, Korea
shchoi@lge.com*

BYONGHWA OH

*Department of Computer Science and Engineering, Sogang University,
35 Baekbeom-ro, Mapo-gu, Seoul 121-742, Korea
mrfive@sogang.ac.kr*

JIHOON YANG*

*Department of Computer Science and Engineering, Sogang University,
35 Baekbeom-ro, Mapo-gu, Seoul 121-742, Korea
yangjh@sogang.ac.kr*

SUNGYONG PARK

*Department of Computer Science and Engineering, Sogang University,
35 Baekbeom-ro, Mapo-gu, Seoul 121-742, Korea
parksy@sogang.ac.kr*

Received 20 March 2012
Accepted 29 June 2013
Published 19 August 2013

We present a new ensemble learning method that employs a set of regional classifiers, each of which learns to handle a subset of the training data. We split the training data and generate classifiers for different regions in the feature space. When classifying an instance, we apply a weighted voting scheme among the classifiers that include the instance in their region. We used 11 datasets to compare the performance of our new ensemble method with that of single classifiers as well as other ensemble methods such as RBE, bagging and Adaboost. As a result, we found that the performance of our method is comparable to that of Adaboost and bagging when the base learner is C4.5. In the remaining cases, our method outperformed other approaches.

Keywords: Ensemble learning; regional classifier; bagging; boosting.

*Corresponding author

1. Introduction

In machine learning, improving the performance of the learned model is one of the most important issues. However, it is not so probable that a particular classifier always yields the best performance on a variety of tasks.[1] This is because each classifier has its own inductive bias, and each dataset has different characteristics. Even in a specific task, there might be instances that are hard to classify by a single classifier (e.g., outliers). Ensemble learning[2] was proposed to overcome this limitation. Ensemble learning does not aim to find a single hypothesis (or classifier) that explains the data best.[a] Instead, it tries to create a new hypothesis by combining a set of hypotheses produced by individual learning algorithms and to classify data more accurately. This is intuitively appealing as humans also look for multiple opinions on various matters (e.g., disease treatments, business plans). The advantage of ensemble learning is usually worth the additional computation of ensemble construction, and ensemble classifiers indeed have been experimentally proven to be more accurate than single classifiers.[3–5]

A variety of methods for ensemble learning have been proposed in the literature.[2–8] In common ensemble learning approaches such as *bagging*[6] and *boosting*,[7] when the ensemble learner classifies a new instance, it does not consider which individual classifier of the ensemble should be applied to the instance. That is, each individual classifier participates in the voting process (either by simple (in bagging) or weighted (in boosting) voting) even though it was not trained for that instance or it does not explain the instance well. If the set of instances (or the regions where they lie) that each classifier is strong at classification is known, the ensemble classifier will be able to perform more precise classification. To this end, we had proposed an ensemble learning algorithm, *RBE* (*Region Based Classifiers*) in our previous research.[8] *RBE* produces a tree of classifiers where each node is a base classifier trained with the samples lying in the region of which the node is in charge. In classification, the classifiers in a particular path in the tree that contain the test pattern in their jurisdictions take part in a simple voting to determine the class label giving preference to classifiers located in lower-level nodes in the case of ties. *RBE* is shown to produce good performance compared to single classifiers and ensemble classifiers of bagging and boosting (See Ref. 8 for detailed descriptions on *RBE* and its performance).

In this paper, we present a new ensemble learning algorithm, *ELRC* (Ensemble Learning of Regional Classifiers). *ELRC* is similar to *RBE* in that it also employs a set of regional classifiers, and it votes on the class label of a test instance among the classifiers that include the instance in their territories. However, *ELRC* makes use of different methods of greedy classifier generation and weighted voting in a bid to obtain accurate classification (See Section 3 for descriptions of the algorithm). The advantage of regional classification of *ELRC* with strong individual classifiers was verified in our experiments.

[a]The term *classifier* and *hypothesis* might have different nuance depending on the context, but both mean the result of the application of a classification algorithm and will be used interchangeably in this paper.

The rest of the paper is organized as follows: Section 2 brings in the issue of the conditions for good ensembles. Section 3 includes detailed descriptions on our new ensemble learning algorithm, *ELRC*. Section 4 explains the data and experimental setup designed to evaluate the performance of *ELRC*, followed by the results of the experiments. Section 5 concludes with summary and discussion of some directions for future research.

2. Conditions for Good Ensembles

A learning algorithm is called *unstable* if "small" changes in the training data lead to significantly different classifiers and relatively "large" changes in accuracy.[6] For instance, the decision tree learning algorithms such as ID3,[9] C4.5[10] are unstable. Such unstable algorithms are known to have a high variance.[11,12] On the contrary, we say an algorithm is *stable* when it has a low variance. Support Vector Machines (SVMs)[13] and Naïve Bayes[14] are examples of stable learning algorithms.

Generally, ensemble learning methods such as bagging and boosting have been known to work well with unstable algorithms though it is independent of the base learner in principle.[6] On the other hand, bagging and boosting did not show any remarkable improvement with naive Bayes or other stable algorithms.[11] In fact, some research even demonstrated the negative effect of SVM-based ensemble learning.[15,16] In addition, Dietterich recommended boosting with such algorithms that create less expressive hypotheses, including the decision tree algorithms.[17] Many of the previous approaches to ensemble learning thus have been focused on the ensemble of decision trees.[3,4,18,19]

The reason why the ensemble of unstable (and less expressive) learning algorithms works well is related to the diversity in the sense that each classifier should have errors in different areas of the input space.[20] As an example, assume that we have an ensemble of three classifiers operated by simple majority voting to combine the predictions of individual classifiers. If those classifiers are not diverse, the behavior of all individual classifiers will be similar without producing improved performance by ensemble learning. There have been many studies on the relation between the diversity and the efficiency of an ensemble.[2,20,21] According to Hansen and Salamon,[20] a necessary and sufficient condition for an ensemble to be more accurate than any of its individual classifiers is that the base classifiers should be accurate and diverse.

Suppose an ensemble classifier is created by combining five individual classifiers, each of which is completely independent of each other and has 70% of classification accuracy. Also suppose a simple majority voting is used to combine the prediction of individual classifiers. For this ensemble classifier to classify a pattern correctly, more than 3 individual classifiers should be correct. We can calculate this probability as follows:

$$\sum_{i=3}^{5}\binom{5}{i}\cdot(0.7)^{i}\cdot(0.5)^{5-i}=0.84. \quad (1)$$

As shown in Eq. (1), the accuracy of this ensemble classifier is approximately 84%, which is 14% higher than that of its individual classifiers. The misclassification rate of this ensemble classifier is approximately 16%, which is almost a half of its individual classifiers' error rate, 30%. If an ensemble classifier is created with 5 classifiers of 90% accuracy, it leads to Eq. (2):

$$\sum_{i=3}^{5} \binom{5}{i} \cdot (0.9)^i \cdot (0.1)^{5-i} = 0.99. \tag{2}$$

As seen in Eqs. (1) and (2), the improvement of the ensemble of 5 classifiers with 70% accuracy is bigger than that with 90% accuracy. However, the latter shows better performance than the former. Therefore, even though the gain in accuracy from forming an ensemble is not as significant, we may as well use more accurate classifiers to construct an even more accurate ensemble classifier.

In conclusion, the base classifiers in ensemble learning need to be both diverse and accurate for high performance classification.

3. Ensemble Learning of Regional Classifiers

As aforementioned, we attempt to design a new ensemble method, *ELRC*, which splits the training data and allocates the splits to different classifiers. *ELRC* can be represented as a tree in which each node corresponds to a classifier that is assigned to a particular region in the feature space. Figure 1 shows the idea and overall structure of *ELRC*.

In *ELRC*, not only the leaf nodes, but also all the internal nodes are classifiers. We use the training instances in a region to generate the corresponding classifier. In addition, each classifier in the tree can be trained by a different learning algorithm. When classifying a new instance, we apply a weighted voting among the classifiers that include the instance in their region.

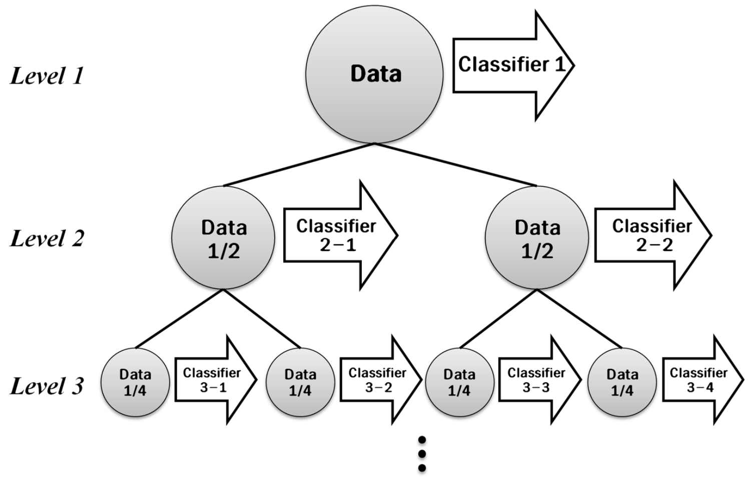

Fig. 1. Overview of *ELRC*.

3.1. *Learning process*

ELRC generates a series of classifiers that corresponds to different regions in the feature space in a recursive manner. A region under consideration is iteratively split into two sub-regions with almost the same number of instances as learning progresses, and the corresponding classifiers are generated using the instances in each sub-region. The learning process of *ELRC* for a sample data set is illustrated in Fig. 2. The process is repeated until at least one of the stopping criteria is met as described below.

Among a predefined group of learning algorithms, we choose the one that produces the best performance in each region. We can consider as many learning algorithms as we wish in order to obtain good performance. In this paper, we consider three algorithms of C4.5,[10] SVM,[13] and Naïve Bayes.[14] For each region, we create three classifiers by the three learning algorithms and select the best one. The three algorithms are chosen since they belong to the set of machine learning algorithms that are most common and widely used nowadays. C4.5 is a decision tree learning algorithm that determines the nodes in the tree based on the *entropy* (as described below) in order to find the most relevant

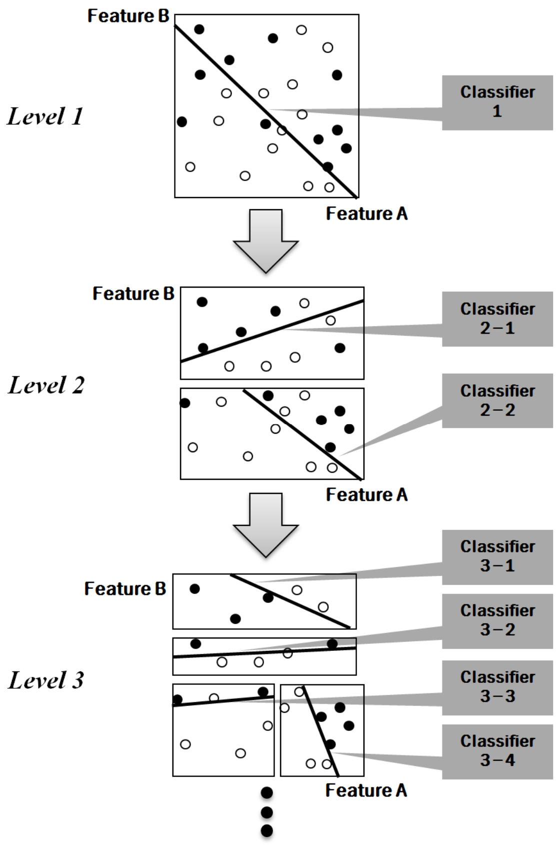

Fig. 2. Illustration of *ELRC* learning process.

feature with respect to the class, and thus to induce the most succinct tree. SVM is a kernel classifier that finds a class-separating hyperplane with the maximal margin. Naïve Bayes computes the conditional probabilities that can be used to compute the posterior probability of the classes, under the strong independence assumption among the features given the class (See the references for more detailed descriptions on the algorithms).

The performance of the learning algorithms is measured through 10-fold cross-validation, where we choose the one with the highest cross-validation accuracy. If there is a tie, the learning algorithm with the highest training accuracy is selected. In case of a tie in training accuracy, we choose the learning algorithm that is selected most frequently by the ancestor nodes when there are ancestor nodes, or follow a predefined priority for the root node. Since we choose the best learning algorithm at each step, we expect to generate accurate classifiers. Meanwhile, the diversity is maintained through utilization of different learning algorithms for different regions.

As shown in Fig. 2, Classifier 1 is generated using the entire data D by the selected learning algorithm. Then the training data D is divided into two regions D_1, D_2 containing nearly the same number of instances. In order to divide the data close to halves, we compute the *Score* of each feature A in regard to the division and select the feature that has the maximum *Score* defined as follows:

$$\begin{aligned} &Entropy(X) = -\sum_{i=1}^{n} p(x_i)\log p(x_i). \\ &Score(D,A) = Entropy(D) - \frac{\big||D_1|-|D_2|\big|}{|D|}\sum_{i=1}^{2}\frac{|D_i|}{|D|}\cdot Entropy(D_i). \end{aligned} \tag{3}$$

Entropy characterizes the impurity of an arbitrary collection of examples.[22] Equation (3) resembles the definition of information gain.[9] Simply put, information gain is the expected reduction in entropy caused by partitioning the examples according to the selected feature.[22] We add $||D_1| - |D_2|| / |D|$ as a weight to the post-split entropy to induce a balanced partition (which is determined by a scan of the distribution of the values of the attribute A), aiming for the maximum reduction of entropy at the same time. If the partitions are imbalanced, the size of the tree (and thus the number of classifiers) can become unnecessarily large. Meanwhile, a small subset with insufficient data might cause an insufficient learning in the classifier. In other words, when the training data is not sufficiently large the probability of selecting the correct hypothesis might decrease. This is because there are more likely to be several hypotheses that have the same maximum accuracy in the hypothesis space of the learning algorithm. Therefore, we need to divide the data as equally as possible as represented in Eq. (3).

We choose the feature with the maximum *Score*, so as to induce subsets of the data with which we can construct accurate classifiers. At the same time, since each classifier is generated from different training data, we can keep the classifiers diverse. Through the repetition of this process on the divided data, succeeding classifiers 2-1 and 2-2 are generated. Similarly, classifiers 3-1, 3-2, 3-3, and 3-4 are generated with the data in the four regions in level 3. *ELRC* keeps repeating this process until at least one of the following stopping conditions are satisfied: if the predefined level is reached, if the entropy of the region is 0, if the classification error in the region is 0, or if the number of instances in the region is less than the threshold. During learning, *ELRC* also needs to store the number of correctly classified instances for a region by the classifier at the node as well as the classifier at the parent node to determine the weight (or significance) of the classifier (This will be explained in the classification process).

The learning process of *ELRC* is summarized in Table 1.

Table 1. Pseudo-code of *ELRC* learning process.

```
function ELRC-Learn(TD, LA, L, MaxL, MinI, default) returns an ELRC-tree
// TD: training instances
// LA: set of learning algorithms
// L: current level
// MaxL: maximum level
// MinI: minimum number of instances
// default: default tree
if (L > MaxL) then return default;
else if ((|TD| < MinI) OR (Entropy(TD) == 0)) then return Mode(TD);
else
    Create the best classifier BC among LA;
    Compute the number of correctly classified instances CI in TD by BC;
    if (|CI| == |TD|) then
        return a new ELRC-tree with root of (BC, CI, null, null, null);    // 100% correct classification
    endif
    Search the splitting point SP of each feature for dividing TD in nearly half;
    Select the feature F that has the highest score;
    for each branch $b_i$ with label $l_i$ of the two branches with respect to SP do
        subTD = {elements of TD under the branch};
        Compute the number of correctly classified instances PCI in subTD by BC;
        tree = a new ELRC-tree with root of (BC, CI, PCI, SP, F);
        subTree = ELRC-Learn(subTD, LA, L+1, MaxL, MinI, Mode(TD));
        Add $b_i$ to tree with $l_i$ and sub-tree subTree;
    endfor
endif
```

3.2. *Classification process*

When *ELRC* classifies a test pattern, the classifiers in the ensemble that include the test pattern in their allocated region vote on the class label. In other words, a pattern is classified by the classifiers in a path of the tree, each of which is in charge of the region that includes the test pattern. Only these classifiers are trained with the patterns in the same regions during the learning process, hence, other classifiers are excluded from voting. As a result, we can assure good performance of such classifiers that participate in the voting process. Figure 3 illustrates the classification process in *ELRC*.

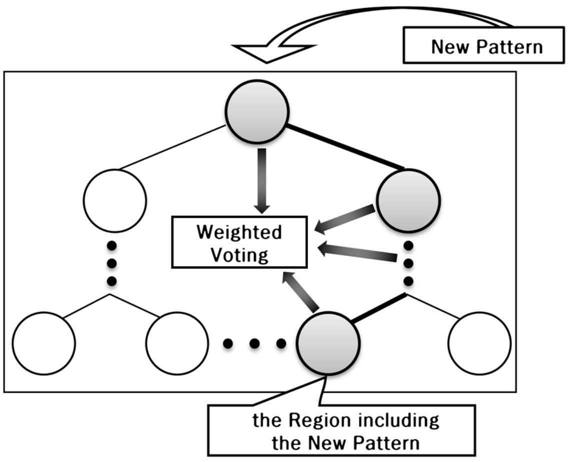

Fig. 3. Illustration of *ELRC* classification process.

Our expectation is that the data subsets would be easier to classify at lower levels of the tree. However, experiments revealed that this was not always true. Cases where the number of errors increased after data splits were often observed. Therefore, we calculate the weight of each classifier according to its accuracy rather than blindly assigning heavier weights to the lower nodes. We calculate the weight of a classifier as follows:

$$Weight(N) = \frac{|I|}{|PI|} \cdot PW. \tag{4}$$

where *N* is a classifier, *I* is the set of instances that are classified correctly by *N* among the instances in the allocated region of *N*, *PI* is the set of instances that are classified correctly by *N*'s parent among the instances in region of *N*, and *PW* is the weight of *N*'s parent. The weight of the classifier in the root node is 1 and each classifier has a different weight according to its classification accuracy. The rationale behind Eq. (4) is that the weight of a classifier is proportional to its relative accuracy to its parent's and the weight of its parent, which emphasizes classifiers with significant improvements from highly weighted parents.

Instead of discrete classifications, we use the class probability of each classifier to determine the final prediction of *ELRC*. The use of class probability increases the number of possible outcomes of classification, which results in higher diversity. The votes of each class are determined according to the weighted class probability. The class with the most votes is decided as *ELRC*'s prediction value. We calculate the vote as Eq. (5)

$$Vote(x, P) = \sum_{i=1}^{L} Weight(N_i) \cdot Class\_Prob(x, P, N_i). \tag{5}$$

and decide the prediction of *ELRC* as Eq. (6)

$$Classification(P) = \arg\max_{i \in C} Vote(i, P). \tag{6}$$

where x is the class, P denotes the pattern, L represents the number of classifiers which vote on the class (i.e., the classifiers on a path of the tree that include P in their jurisdiction), N_i is the classifier which votes on the class, *Class_Prob*(x, P, N_i) is the class probability that classifier N_i predicts pattern P as class x, and C stands for the set of classes.

Note that overfitting can be caused by the classifier that does not rely on the distribution of the whole data but on the distribution of data in the regionally split areas instead. Therefore, classifiers in the lower nodes have high chance of overfitting. The classifiers in the upper nodes in the *ELRC*-tree can minimize the risk of overfitting because they provide more generalization than the classifiers in the lower nodes. That is why the upper nodes participate in voting as well as the lower nodes.

The classification process of *ELRC* is summarized in Table 2.

Table 2. Pseudo-code of *ELRC* classification process.

```
function ELRC-Classify(X, Tree, MaxL) returns a Classification
// X: instance to classify
// Tree: ELRC-tree
// MaxL: maximum level
for $i = 1$ to $C$ do    // C: number of classes
    $vote_i = 0$;
endfor
$index = W = 1$;    // index and weight of root node
for $i = 1$ to $C$ do
    $vote_i$ = class probability of $i$-th class predicted at root node × $W$;
endfor
for $i = 2$ to $MaxL$ do    // follow the path from node on level 2 to the last node
    $parentW = W$;
    if ($\exists$ node with ($BC$, $CI$, $PCI$, $SP$, $F$)) then
    // BC: best classifier
    // CI: number of correctly classified patterns by BC at current  node
    // PCI: number of correctly classified patterns by BC at parent  node
    // SP: splitting point
    // F: feature used in split
        if ($value(F(X)) < SP$) then $index$ = index of left child node;
        else $index$ = index of right child node;
        endif
        $W = (CI / PCI) * parentW$;    // weight of node index
        for $i = 1$ to $C$ do
            $vote_i$ += class probability of $i$-th class predicted at node index × $W$;
        endfor
    endif
endfor
return $\mathrm{argmax}_i(vote_i)$;
```

3.3. *Analysis of ELRC*

Splitting the data in *ELRC* has the following two significant implications. First, as it goes on splitting the data, the classifiers become more accurate. This is because a split tends to simplify the classification task with less number of training instances. Second, the classifiers generated from the data before a split can supplement the classifiers generated after the split. Even though the classifiers in the lower nodes have less errors upon the training data, not all the instances that are correctly classified by the classifiers in the upper nodes are always correctly classified by the classifiers in the lower nodes as well. This is related to the diversity that each classifier has errors in different parts of the feature space. These aspects of improved accuracy and diversity in the classifiers produced by *ELRC* become the basis for efficient ensemble learning and are verified in the following theorems similar to the ones shown in Ref. 8.

Theorem 1. Assume that for given training data D_i, the hypothesis produced by the best learning algorithm *BC* in *ELRC* is h_i. Also assume that hypotheses h_0, h_1, h_2 are produced by *BC* for data D_0 and its split data D_1 and D_2, respectively. If the number of errors of h_0, h_1, h_2 upon their own training data are e_0, e_1, e_2 respectively, $e_0 \geq e_1 + e_2$ (Condition: The learning algorithm *BC* chooses the hypothesis which has the least number of training errors upon the data from the hypothesis space it can represent).

Proof (By contradiction)**.** Let e_{01} and e_{02} be the errors made by h_0 for the instances in D_1 and D_2, respectively. Clearly $e_0 = e_{01} + e_{02}$. In order to prove $e_0 \geq e_1 + e_2$, it is sufficient to show $e_{01} \geq e_1 \wedge e_{02} \geq e_2$.

In the hypothesis space *BC* can express, the set of hypotheses generating the same value as h_0 upon D_1 is defined as H_{01}. Then $h_0 \in H_{01}$ is evident since the hypotheses in H_{01} can have any result value upon D_2 as long as they have the same value as h_0 upon D_1. Thus, every hypothesis in H_{01} has e_{01} errors on D_1.

Suppose $e_{01} < e_1$. If H_{01} belongs to the hypothesis space *BC* can express, by the given condition *BC* should choose the hypothesis with e_{01} errors (i.e., h_0) rather than the hypothesis with e_1 errors (i.e., h_1). However, *BC* did choose h_1 over h_0 on D_1. So H_{01} does not belong to the hypothesis space. This is a contradiction because *BC* in fact chose h_0 belonging to H_{01} when D_0 was given. Likewise, it can be also proved that $e_{02} < e_2$ is false. Consequently neither $e_{01} < e_1$ nor $e_{02} < e_2$ is true. Therefore $e_{01} \geq e_1 \wedge e_{02} \geq e_2$ is true. □

Theorem 2. Assume that h_i is the hypothesis produced from a given data D_i by the best learning algorithm *BC* in *ELRC*. Also assume that hypotheses h_0, h_1, h_2 are produced by *BC* for data $D_0 = \{(X_j, y_j)\}$ $(j = 1,\dots, m)$ and its split data D_1 and D_2, respectively. Then, for every X_j such that $X_j \in D_0$, $h_0(X_j) = y_j$ $(j = 1,\dots, m)$, it is not always true that for $X_j \in D_1$, $h_1(X_j) = y_j$ and for $X_j \in D_2$, $h_2(X_j) = y_j$.

Proof (By an example)**.** Figure 4 shows a two-dimensional XOR data classified by a two-level *ELRC* with Naïve Bayes classifier. It was proved that there existed certain instances which the classifiers after the split were not able to classify while the classifier before the split could (Such instances are marked by red circles). □

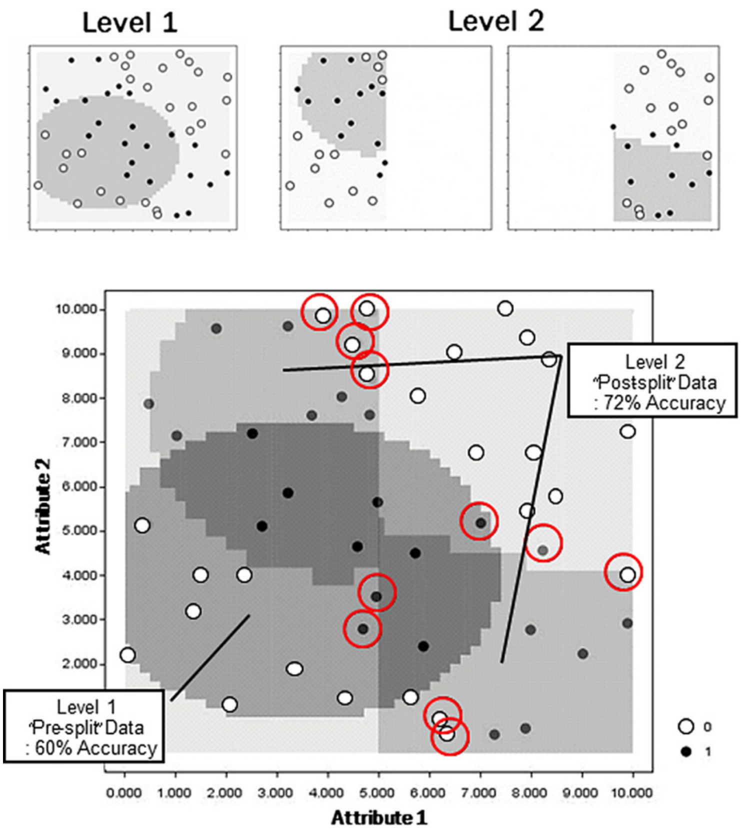

Fig. 4. (Color online) Decision boundaries defined by classifiers in the upper and lower levels.

3.4. *Time complexity of ELRC*

The time needed for learning of *ELRC* can be divided into the time for splitting the data and the time for generating a classifier from each split by a learning algorithm. First, the time for splitting the data is given by

$$S(n) = O(d \cdot n) + O(d \cdot n) + O(n) = O(d \cdot n). \tag{7}$$

where n is the number of instances and d is the number of features. The time for data splitting consist of the time for finding the split point of each feature, the time for calculating the score (as in Eq. (3)), and the time for splitting the data after choosing the feature with the highest score, which are $O(d \cdot n)$, $O(d \cdot n)$ and $O(n)$, respectively. Consequently the total time complexity for splitting the data is $O(d \cdot n)$.

Now suppose the time for generating a classifier by a learning algorithm is $L(n)$. The number of instances decreases by half and the number of nodes increases double. Thus the total time needed for *ELRC* with N levels is

$$E(N) = \sum_{i=1}^{N-1}\left(2^{i-1}\cdot S\left(\frac{n}{2^{i-1}}\right)\right) + \sum_{i=1}^{N}\left(2^{i-1}\cdot L\left(\frac{n}{2^{i-1}}\right)\right). \tag{8}$$

In Eq. (8), the first part can be expressed as follows:

$$\begin{aligned}\sum_{i=1}^{N-1}\left(2^{i-1}\cdot S\left(\frac{n}{2^{i-1}}\right)\right) &= \sum_{i=1}^{N-1}\left(2^{i-1}\cdot O\left(d\cdot\frac{n}{2^{i-1}}\right)\right) \le C\cdot d\cdot n\cdot\sum_{i=1}^{N-1}\left(2^{i-1}\cdot\frac{1}{2^{i-1}}\right)\\ &= C\cdot d\cdot n\cdot(N-1). \quad (C \text{ is constant})\end{aligned} \tag{9}$$

In order to calculate the time complexity of $L(n)$, assume that we use the decision tree algorithm as the base learning algorithm. The complexity of decision tree algorithm is known as $O(d\cdot n\cdot(\log n)^2)$.[23] The rest of the Eq. (8) can be simplified:

$$\begin{aligned}\sum_{i=1}^{N-1}\left(2^{i-1}\cdot L\left(\frac{n}{2^{i-1}}\right)\right) &= \sum_{i=1}^{N-1}\left(2^{i-1}\cdot O\left(d\cdot\frac{n}{2^{i-1}}\cdot\left(\log\frac{n}{2^{i-1}}\right)^2\right)\right)\\ &\le C\cdot d\cdot n\cdot\sum_{i=1}^{N-1}\left(2^{i-1}\cdot\frac{1}{2^{i-1}}\cdot\left(\log\frac{n}{2^{i-1}}\right)^2\right) = C\cdot d\cdot n\cdot\sum_{i=1}^{N-1}\left(\log\frac{n}{2^{i-1}}\right)^2\\ &\le C\cdot d\cdot n\cdot(N-1)\cdot(\log n)^2. \quad (C \text{ is constant})\end{aligned} \tag{10}$$

According to Eqs. (9) and (10), (8) can now be finalized:

$$\begin{aligned}&\text{Time for learning } ELRC \text{ with } N \text{ levels}\\ &= O(d\cdot n\cdot N) + O(d\cdot n\cdot N\cdot(\log n)^2) = O(N\cdot d\cdot n\cdot(\log n)^2).\end{aligned} \tag{11}$$

Now we know that the time complexity of *ELRC* (with N levels) increases linearly with respect to the complexity of the base learning algorithm and N. This is because the number of instances used for learning decreases by half of the parent node's while the number of generated classifiers increases in the rate of $2^N - 1$ as N increases. This is different from other ensemble methods such as bagging or boosting in that such methods generate one classifier at each iteration and the amount of data used for learning does not decrease. In other words, the time complexity of bagging or boosting increases linearly according to the number of classifiers created because the amount of learning data used by each classifier remains constant. Thus *ELRC* has a significant advantage in time complexity compared to bagging and boosting if they combine the same number of classifiers.

4. Experiments

4.1. *Experimental setup*

In our experiments, we used 11 data sets from the UCI Machine Learning Data Repository.[24] Table 3 summarizes the datasets. We chose Naïve Bayes, C4.5 and SMO (which is the sequential minimal optimization algorithm for training a support vector classifier)[25] as the base learning algorithms, and gave a priority among the algorithms in the order of SMO, Naïve Bayes, and C4.5 for resolving ties.

Table 3. Datasets.

Dataset	# of Instances	# of Classes	# of Attributes
Abalone	4177	28	8
Ecoli	336	8	7
Glass	213	6	9
Hypothyroid	215	3	5
Ionosphere	351	2	34
Page-Blocks	5473	5	10
Segment	2310	7	19
Spambase	4601	2	57
SPECTF	267	2	44
Vehicle	846	4	18
Waveform	5000	3	40

In order to measure the performance of *ELRC* in relation to existing methods, a single classifier used in *ELRC* as the base learning algorithm, an Adaboost[23] ensemble classifier, a bagging ensemble classifier, and *RBE* were compared in terms of the classification accuracy. We adopted the same algorithms used in *ELRC* as the base learning algorithms of the other ensemble methods. In *ELRC*, we generated classifiers with the three base learning algorithms to select the best one. Thus we generated 21 classifiers in order to learn *ELRC* having three levels (and thus 7 nodes) with the threshold (for the minimum number of instances in a region) of 10. For a fair comparison, boosting and bagging also generated 21 individual classifiers. As in *ELRC*, *RBE* was also expanded up to three levels. Weka 3.5.7,[26] the open data mining software, was used for the base learning algorithms, Adaboost, and bagging.

4.2. *Results*

Table 4 shows the average accuracies of single classifiers and the ensemble methods such as bagging, Adaboost, *RBE*, and *ELRC* that use SMO, Naïve Bayes and C4.5 as the base learning algorithms (Ada means Adaboost and Bagg means bagging). 10-fold cross-validation was used to measure the performance. The most accurate classifiers among various methods are shown in boldface.

First, in comparison to single classifiers, *ELRC* always produced higher accuracy of the base learning algorithm. This verifies significantly improved performance of *ELRC* over single classifiers.

ELRC was also generally more accurate than bagging and Adaboost with SMO or Naïve Bayes as the base learning algorithm. When either SMO or Naïve Bayes were used, Adaboost and bagging failed to improve the performance of the single classifier. The poor performance of Adaboost and bagging is due to the fact that SMO and Naïve Bayes are stable learning algorithms. On the other hand, *ELRC* holds more diversity in terms of the base learners than Adaboost and bagging because each classifier in the *ELRC* can use different learning algorithms. Therefore, *ELRC* is more accurate than Adaboost and bagging on datasets that are best classified by SMO or Naïve Bayes single classifiers.

Table 4. Accuracy comparison of *ELRC* and other methods.

	SMO				Naïve Bayes				C4.5				
Dataset	Single	Ada	Bagg	RBE	Single	Ada	Bagg	RBE	Single	Ada	Bagg	RBE	ELRC
Abalone	25.3	25.3	25.6	26.7	23.8	23.8	23.6	21.4	20.9	22.2	24.5	20.8	**27.1**
Ecoli	84.2	86.3	83.6	85.7	86.3	86.2	86.3	**86.6**	70.3	82.4	84.5	85.4	**86.6**
Glass	53.9	58.1	55.2	66.1	47.5	47.9	50.8	57.8	69.4	**77.4**	76.6	72.5	74.2
Hypothyroid	89.9	95.8	89.8	94.9	96.8	95.8	97.2	**97.2**	94.9	94.4	93.5	94.4	**97.2**
Ionosphere	88.0	87.8	88.0	88.9	82.9	91.8	82.6	89.8	88.1	**93.2**	91.2	90.1	92.3
Page-Blocks	92.9	92.9	93.4	95.2	90.2	90.2	90.0	90.4	97.1	97.2	**97.4**	97.1	97.2
Segment	92.9	93.2	92.9	93.5	79.7	79.7	80.0	89.0	96.8	**98.3**	97.5	97.4	97.5
Spambase	90.4	90.8	90.8	91.3	79.5	79.5	79.7	79.9	93.0	**95.4**	94.2	92.5	93.7
SPECTF	79.4	76.4	79.4	79.8	67.9	68.5	70.4	72.8	74.5	80.2	79.8	78.7	**81.3**
Vehicle	74.6	74.4	74.9	76.6	46.3	46.3	46.5	67.1	75.0	76.9	74.3	74.3	**77.3**
Waveform	**86.6**	**86.6**	**86.6**	86.5	80.0	80.0	80.1	82.8	75.4	83.5	82.4	78.5	**86.6**
Average	78.0	78.9	78.2	80.5	71.0	71.8	71.6	75.9	77.8	81.9	81.4	80.2	**82.8**

However, *ELRC* was comparable to bagging and Adaboost with C4.5 as the base learning algorithm. *ELRC* was actually composed of 7 classifiers, while Adaboost and bagging were composed of 21 classifiers. When classifying a pattern, *ELRC* used 3 classifiers, while Adaboost and bagging used as many as 21 classifiers. Thus, Adaboost and bagging can make more diverse predictions of classifiers than *ELRC*. Moreover, Adaboost and bagging apply sampling for each classifier and thus learn on possibly very different training data, while this is not the case in *ELRC* because the dataset which was assigned to a child node must be assigned to its parent (and ancestor) nodes as well. Thus, Adaboost and bagging yielded more accurate results than *ELRC* in some datasets when C4.5 was used as the base learning algorithm.

In comparison to *RBE*, *ELRC* showed better performance even though *RBE* also improved the performance of single classifiers. This is because each classifier in *ELRC* can employ different learning algorithms at each node with the best performance. Moreover, *ELRC* calculates the weight of each classifier according to the classification accuracies of parent and child nodes, rather than simply giving heavier weights (or preferences) to the lower nodes. *ELRC* also computes class probabilities instead of discrete classifications. Therefore, *ELRC* can hold more diversity and use more accurate base learners than *RBE*.

Finally, *ELRC* produced the best averaged performance over all other approaches (Single, Adaboost, bagging, *RBE*) regardless of the base learning algorithms.

5. Conclusion

We proposed a new ensemble method — ensemble of regional classifiers, *ELRC*. Compared with the existing methods such as bagging and boosting, our algorithm demonstrated outstanding performance when combined with stable learning algorithms as the base learner. *ELRC* was also comparable to bagging and Adaboost with unstable algorithms like C4.5 as the base learner. In comparison to our previous ensemble learning algorithm *RBE*, *ELRC* also showed improved performance.

ELRC's automated determination of the best learning algorithm frees the user from the selection of the base algorithm. The user is only responsible for deciding the candidate group for the base learning algorithms. This is an even stronger merit compared to other approaches that are dependent on ad hoc, arbitrary, and possibly far from optimal parameter settings (e.g., temporal and spatial scale parameters used in Refs. 27–29). We expect that data splitting would induce subsets easier to be classified and generate different training data for regional classifiers. This will lead to the construction of an ensemble with accurate base learners. Meanwhile, in the unfortunate cases where the correctness of the classifiers drops by the splits, we lower the weight of the regional classifier in order to reduce the influence of such classifiers. As well, we are able to create a variety of classifiers from different data subsets.

When classifying a new instance, *ELRC* utilizes the classifier allocated to the region that includes the same instance, and its ancestors which are also trained with the patterns in the region. The rationale behind this idea is to obtain high classification accuracy as well as to prevent overfitting, which are theoretically proved.

In its practical use, the accuracy of *ELRC* may start decreasing from a certain level. So it needs to be examined further to develop a well-defined method to determine the level of the tree for optimal performance. For example, methods that remove unnecessary classifiers from *ELRC*, similar to the pruning method for decision tree classifiers, can be a remedy for this problem. Also, additional state-of-the-art learning algorithms can be considered as base learners. Finally, *ELRC* can be more thoroughly evaluated thru comparative studies with other regionalized approaches (e.g., spatio-temporal methods[27–29]). These are left as topics for future research.

Acknowledgments

This work was supported by Basic Science Research Program (2013R1A1A2012502) through the National Research Foundation of Korea (NRF) to Jihoon Yang and by Convergence Information Technology Research Center support program (NIPA-2013-H0301-13-3006) through the National IT Industry Promotion Agency (NIPA) to Sungyong Park, funded by the Ministry of Science, ICT&Future Planning (MSIP).

References

1. D. H. Wolpert, The lack of *a priori* distinctions between learning algorithms, *Neural Computation* **8**(7) (1996) 1341–1390.

2. T. G. Dietterich, Ensemble method in machine learning, in *Proc. 1st Int. Workshop on Multiple Classifier Systems* (Cagliari, Italy, 2000), pp. 1–15.
3. E. Bauer and R. Kohavi, An empirical comparison of voting classification algorithm: bagging, boosting, and variants, *Machine Learning* **36**(1–2) (1999) 105–142.
4. T. G. Dietterich, An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization, *Machine Learning* **40**(2) (2000) 139–157.
5. D. Optiz and R. Maclin, Popular ensemble methods: an empirical study, *J. Artificial Intelligence Research* **11** (1999) 169–198.
6. L. Breiman, Bagging predictors, *Machine Learning* **24**(2) (1996) 123–140.
7. Y. Freund and R. Schapire, Experiments with a new boosting algorithm, in *Proc. 13th Int. Conf. Machine Learning* (Bari, Italy, 1996), pp. 148–156.
8. S. Choi, B. Lee and J. Yang, Ensembles of region based classifiers, in *Proc. 7th Int. Conf. Computer and Information Technology* (Fukushima, Japan, 2007), pp. 41–46.
9. J. Quinlan, Induction of decision tree, *Machine Learning* **1**(1) (1986) 81–106.
10. J. Quinlan, *C4.5: Programs for Machine Learning* (San Mateo, CA: Morgan Kaufmann, 1993).
11. L. Breiman, Bias, variance, and arcing classifiers, Technical Report TR 460 (UC Berkeley, CA, 1996).
12. R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd edn. (Wiley-Interscience, New York, 2000).
13. V. Vapnik, *The Nature of Statistical Learning Theory* (Berlin: Springer-Verlag, 1995).
14. P. Domingos and M. Pazzani, On the optimality of the simple Bayesian classifier under zero-one loss, *Machine Learning* **29** (1997) 103–137.
15. I. Buciu, C. Kotropoulos, and I. Pitas, Combining support vector machines for accuracy face detection, in *Proc. 8th Int. Conf. Image Processing* (Thessaloniki, Greece, 2001), pp. 1054–1057.
16. T. Evgeniou, L. Perez-Breva, M. Pontil, and T. Poggio, Bound on the generalization performance of kernel machine ensembles, in *Proc. 17th Int. Conf. Machine Learning* (Stanford, CA, USA, 2000), pp. 271–278.
17. T. G. Dietterich, Ensemble Learning, *The Handbook of Brain Theory and Neural Networks*, 2nd edn. (Cambridge, MA: The MIT Press, 2002), pp. 405–408.
18. J. Friedman, T. Hastie and R. Tibshirani, Additive Logistic Regression: A statistical view of boosting, *Annals of Statistics* **28**(2) (2000) 337–374.
19. J. Quinlan, Bagging, boosting, and C4.5, in *Proc. 13th National Conf. Artificial Intelligence* (Portland, OR, USA, 1996), pp. 725–730.
20. L. Hansen and P. Salamon, Neural network ensembles, *IEEE Trans. on Pattern Analysis and Machine Intelligence* **12** (1990) 993–1001.
21. L. Kuncheva and C. Whitaker, Measures of diversity in classifier ensembles, *Machine Learning* **51** (2003) 181–207.
22. T. M. Mitchell, *Machine Learning* (McGraw-Hill, New York, 1997).
23. Y. Freund and R. Schapire, A decision-theoretic generalization of online learning and an application to boosting, *J. Comp. and System Sci.* **55** (1997) 119–139.
24. A. Frank and A. Asuncion, *UCI Machine Learning Repository* [http://archive.ics.uci.edu/ml], Irvine, CA, University of California, School of Information and Computer Science, 2010).
25. J. Platt, Fast training of support vector machines using sequential minimal optimization, *Advances in Kernel Methods: Support Vector Learning* (Cambridge, MA: The MIT Press, 1999), pp. 185–208.
26. I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd edn. (San Francisco, CA: Morgan Kaufmann, 2005).

27. D. Fink, W. M. Hochachka, B. Zuckerberg, D. W. Winkler, B. Shaby, M. A. Munson, G. Hooker, M. Riedewald, D. Sheldon, and S. Kelling, Spatiotemporal exploratory models for broad-scale survey data, *Ecological Applications* **20**(8) (2010) 2131–2147.
28. A. O. Finley, Comparing spatially-varying coefficients models for analysis of ecological data with non-stationary and anisotropic residual dependence, *Methods in Ecology and Evolution* **2** (2011) 143–154.
29. A. S. Fotheringham, C. Brunsdon, and M. Charlton, *Geographically Weighted Regression: The Analysis of Spatially Varying Relationships*, (Wiley, 2002).