2016 23rd International Conference on Pattern Recognition (ICPR)
Cancún Center, Cancún, México, December 4-8, 2016

Incremental Construction of Rule Ensembles using Classifiers Produced by Different Class Orderings

Olcay Taner Yıldız[*], Aydın Ulaş[†]
[*]Department of Computer Engineering, Işık University, Istanbul, Turkey
[†]ARGELA A.Ş., Network & Services Innovation Center, Istanbul, Turkey
Email: olcaytaner@isikun.edu.tr, aydin.ulas@argela.com.tr

Abstract—**In this paper, we discuss a novel approach to incrementally construct a rule ensemble. The approach constructs an ensemble from a dynamically generated set of rule classifiers. Each classifier in this set is trained by using a different class ordering. We investigate criteria including accuracy, ensemble size, and the role of starting point in the search. Fusion is done by averaging. Using 22 data sets, floating search finds small, accurate ensembles in polynomial time.**

Index Terms—**Ensemble construction, Rule sets**

I. Introduction

It is well-known that combining classifiers based on different learning algorithms [1], [2] gives better results than single classifiers. Each algorithm makes a different assumption and hence errs on different instances and by combining these, the overall error can be decreased.

To decrease error, classifiers in the ensemble should be carefully chosen. Combination through averaging reduces variance but the error decreases if only the decrease in variance is greater than the increase in bias. This brings the necessity to use classifiers that contribute to accuracy, hence classifiers performing poorly should be weeded out. Unfortunately, it is not possible to train an unlimited number of classifiers and combine them in the most suitable way. Each additional classifier increases space and computational complexity added to the concomitant increase in cost of extracting/sensing the features for the classifier; thus a new classifier should be avoided if it is redundant. Therefore, methods have been proposed to choose a small subset from a large set of candidate classifiers. Since there are 2^L possible subsets of L classifiers, it is impossible to try for all possible subsets unless L is small, and various methods have been proposed to get a reasonable subset of size $m < L$ in reasonable time.

Ensemble construction by subset selection can also be considered as an optimization problem [3], [4] and methods proposed in the literature correspond to different search strategies in optimization: There are "forward" algorithms which are incremental and add classifiers if the addition improves the criterion to be optimized. There are "backward" search methods which prune from a large set if the removal does not decrease the criterion. There are also "floating" methods which do both, as well as ones that use genetic algorithms whose operators allow both addition and deletion. All of these methods require the classifiers to be already trained which brings additional cost if they are redundant.

Rule induction algorithms learn rules to separate a *positive* class from a set of classes. In each iteration, a class is selected to be a *positive* class, and the instances (mostly belonging to the *positive* class) satisfying the rules are separated from the training set. The ordering of classes (π) fed to the algorithm is selected heuristically and may not be optimal in terms of error and/or complexity.

Each ordering fed to the algorithm results in a different classifier, and therefore will have a different inductive bias. A rule classifier with class ordering π may be accurate for some classes but not all. Therefore, it is best to use that rule classifier with other rule classifiers which are good at other classes.

In this paper, we discuss and evaluate a new **R**ule **EN**semble (REN) construction approach, where we use different rule learning algorithms starting with different class orderings as different base classifiers. We incrementally construct an ensemble of classifiers using floating search discussed above. On 22 data sets, we compare the performance of our ensemble with respect to (i) accuracy, (ii) ensemble size (number of base classifiers), and (iii) space size (number of distinct candidate ensembles searched).

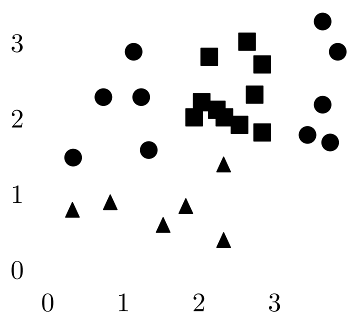
The paper is organized as follows: In Section II we briefly discuss rule induction and a well-known rule inducer, namely Ripper. In Section III we introduce our ensemble construction algorithm. We give the details of our experiments in Section IV and results in Section V. We conclude in Section VI.

II. Rule induction

In rule induction, the main objective is to extract rule sets from data. A rule set is typically composed of an ordered list of rules, where a rule is composed of a conjunction of a list of conditions [5]. Depending on the type of the input attribute, the conditions are of the form

- $x_i = v$: if x_i is discrete
- $x_i \leq \theta$ or $x_i > \theta$: if x_i is continuous

A rule is said to *cover* an instance, if that instance satisfies all conditions in that rule. Each rule is associated with a class label, and class label of the first covering rule is assigned to an instance. If none of the rules cover an instance, the default class label is assigned to that instance. An example rule set composed of three rules is given below.



If (x_1 < 3) **and** (x_2 < 1)
  Then class = ▲
Else
  If (x_1 > 1.5) **and** (x_2 < 1.5)
    Then class = ▲
  Else
    If (x_1 < 1.5)
      Then class = ●
    Else
      If (x_1 > 3)
        Then class = ●
      Else class = ■

Fig. 1. For a specific class ordering, separation of data and learned ruleset.

If x_2 = 0 **and** x_4 = 0 **Then** C_1
Else
  If x_1 = 1 **and** x_3 = 0 **and** x_5 = 1 **Then** C_1
  Else
    If x_1 = 0 **Then** C_1
    Else C_0

In rule induction, when a rule is learned for class C_i, the covered examples are removed from the training set. This procedure proceeds until no examples remain from class C_i in the training set. If we have two classes, we separate positive class from negative class. But if we have $K > 2$ classes, we use a class ordering π, and follow one-vs-all strategy to separate every class C_i from classes C_j coming after C_i in the ordering π. Well known rule induction algorithms are C4.5Rules [6], PART [7], CN2 [8] and Ripper [9].

In Figure 1, we see an example case, where a rule inducer first learnes rules to separate class ▲ from both classes ● and ■, then learnes rules to separate class ● from class ■.

An example rule learning algorithm Ripper, learns rules from scratch starting from an empty rule set. It has two phases: In the first phase, it builds an initial set of rules, one at a time, and in the second phase, it optimizes the rule set m times [9].

The simplified pseudocode for learning ruleset from examples using Ripper is given in Figure 2. When there are $K > 2$ classes, the classes of the dataset are given with permutation π (Line 1). For each class π_p, its examples are considered as positive and the examples of the remaining classes $\pi_{p+1}, ..., \pi_K$ are considered as negative (Line 4). Rules are grown (Line 8), pruned (Line 9) and added (Line 10) one by one to the rule set. If there are no remaining positive examples (Line 6) rule adding stops. After learning a rule set, it is optimized twice (Line 13).

```
1  Ruleset Ripper($D$, $\pi$)
2    $RS$ = {}
3    for $p$ = 1 to $K-1$
4      Pos = $\pi_p$, Neg = $\pi_{p+1}, \ldots, \pi_K$
5      $RS_p$ = {}
6      while $D$ contains positive samples do
7        Divide $D$ into Grow set $G$ and Prune set $P$
8        $r$ = GrowRule($G$)
9        PruneRule($r$, $P$)
10       $RS_p$ = $RS_p$ + $r$
11       Remove examples covered by $r$ from $D$
12     for $i$ = 1 to 2
13       OptimizeRuleset($RS_p$, $D$)
14     $RS$ = $RS$ + $RS_p$
15   return $RS$
```

Fig. 2. Simplified pseudocode for learning a ruleset using Ripper on dataset D according to class ordering π

III. Proposed Approach: Rule Ensemble

A rule classifier RC_i with class ordering π_i may be accurate for some classes but not all. For that reason, it is best to use RC_i with other rule classifiers which are good at other classes. Each rule classifier with a different class ordering has a certain inductive bias, which may hold for some classes but not all. For example, one of the classes C_j may be easily separable from the others but another class C_k may be not. In that case, the output of the rule classifier which has C_j first in the ordering may therefore be included in the ensemble. But for the second class C_k, one may need a combination of rule classifiers to create a more complex discriminant. Certain classes may be handled by a single rule classifier, i.e., the output of one rule classifier suffices, but more classifiers are combined for more complex classes.

Our proposed algorithm, namely Ren, views ensemble construction problem as an optimization problem, namely as a search in the state space of all possible ensembles. The search space contains all possible ensembles of rule classifiers. In our case, the problem is much more complex, not only the state space of ensembles is extremely huge, but also the number of possible rule classifiers to put in an ensemble is huge.

Although the number of rule classifiers is finite, it is not possible to train/validate all rule classifiers for all $K!$ distinct orderings. Since there are $K!$ distinct orderings, there are also $2^{K!}$ possible ensembles. So there is need for a strategy which finds a good ensemble visiting only a small part of the search space, that is without training all rule classifiers.

Figure 3 shows the pseudocode for constructing an ensemble of rulesets using Ren on dataset D. In the beginning of the search, the algorithm generates m random permutations as class orderings (Line 3). For each of these initial random

```
1  Ensemble Ren($D$, $m$)
2    $S$ = {}
3    $\pi$ = Generate an array of $m$ random permutations
4    **for** $i$ = 1 **to** $m$
5      $RS_i$ = Ripper($D$, $\pi_i$)
6      $S[i].RS$ = $RS_i$
7      $S[i].\pi$ = $\pi_i$
8    $bestError$ = Error($S$)
9    $improved$ = **true**
10   **while** $improved$
11     $improved$ = **false**
12     **for** $i$ = 1 **to** size($S$)
13       $\pi$ = NeighborPermutations($S[i].\pi$)
14       **for** $j$ = 1 **to** $K - 1$
15         $RS_j$ = Ripper($D$, $\pi_j$)
16         $E$ = Error($S \cup (RS_j, \pi_j)$)
17         **if** $E < bestError$
18           $improved$ = **true**
19           $bestError$ = $E$
20           $bestCandidate$ = $(RS_j, \pi_j)$
21           $bestOperation$ = Add
22     **for** $i$ = 1 **to** size($S$)
23       $E$ = Error($S - (S[i].RS, S[i].\pi)$)
24       **if** $E < bestError$
25         $improved$ = **true**
26         $bestError$ = $E$
27         $bestCandidate$ = $(S[i].RS, S[i].\pi)$
28         $bestOperation$ = Remove
29     **if** $improved$
30       **if** $bestOperation$ = Add
31         $S = S \cup bestCandidate$
32       **else**
33         $S = S - bestCandidate$
34   **return** $S$
```

Fig. 3. Pseudocode for constructing an ensemble of rulesets using REN on dataset D with m random base classifiers at the beginning of the search.

orderings, the algorithm trains Ripper (Line 5) and combines all the rulesets produced to construct an initial ensemble (Lines 6-7). The estimated generalization error of this initial ensemble will be the starting point of the floating search (Line 8).

In floating search, there are two basic candidate operations: adding a base classifier to an ensemble (Lines 12-21) and removing a base classifier from an ensemble (Lines 22-28). At each iteration, we extract the best of these candidate operations and if the best one improves the generalization error of the ensemble (Line 29), we apply it (Lines 31, 33). If none of these operations improves the ensemble (Line 10), we return it (Line 34).

In floating search, all base classifiers are possible candidates to add into an ensemble. But in our case, as explained above, running all $K!$ base classifiers is nearly impossible. For that reason, we only select a subset of base classifiers for introducing into the ensemble. We use a neighborhood function to generate class orderings of those subset of classifiers (Line 13).

The neighborhood function on the permutation space that allows generating $K - 1$ neighbor permutations from one permutation. Let say we have the permutation $\pi_0 = C_1C_2C_3 \ldots C_{K-1}C_K$. $K - 1$ neighbor permutations of π are the following permutations: $\pi_1 = C_2C_1C_3 \ldots C_{K-1}C_K$, $\pi_2 = C_1C_3C_2 \ldots C_{K-1}C_K$, $\pi_3 = C_1C_2C_4 \ldots C_{K-1}C_K, \ldots,$ $\pi_{K-1} = C_1C_2C_3 \ldots C_KC_{K-1}$.

For each base classifier in the current ensemble (Line 12), we generate $K - 1$ neighbor permutations using the class ordering of that base classifier (Line 13). Then, taking each neighbor permutation as a class ordering, we train Ripper (Line 15) and check if the generalization error of the new ensemble (Line 16), including the new base classifier trained, is less than the current best generalization error (Line 17). If the generalization error of the new ensemble improves, we set the new best generalization error (Line 19) and note the current optimal operation (Lines 20-21).

As the second basic operation, for each base classifier in the current ensemble (Line 22), we remove it and check if the generalization error of the new ensemble (Line 23), is less than the current best generalization error (Line 24). If the generalization error of the new ensemble improves, we set the new best generalization error (Line 26) and note the current optimal operation (Lines 27-28).

IV. EXPERIMENTAL DETAILS

A. Datasets

We use a total of 22 datasets where 17 of them (*balance, car, cmc, dermatology, hayesroth, iris, led7, mfeatmor, ocr, page-block, segment, splice, tae, vehicle, wave, wine, and yeast*) are from UCI repository [10] and 5 (*braintumor1, braintumor2, leukemia1, leukemia2, and srbct*) are bioinformatic datasets [11] (See Table I for details of the datasets).

B. Base classifiers

We use Ripper with different class orderings as base classifiers.

C. Division of training, validation, and test sets

Our methodology is as follows: A given dataset is first divided into two parts, with $1/3$ as the test set, *test*, and $2/3$ as the training set, *train-all*. The training set, *train-all*, is then resampled using 5×2 cross-validation (cv) [12] where 2-fold cv is done five times (with stratification) and the roles swapped at each fold to generate ten training and validation folds, tra_i, val_i, $i = 1, \ldots, 10$. tra_i are used to train the base classifiers. These ten trained models are tested on the same *test* and we have ten $test_i$ accuracy results.

D. Compared Ensembles

The following ensembles or classifiers are generated and compared:

- BEST: We order the base classifiers in terms of accuracy and use the best of them.

TABLE I
Details of the datasets. d: Number of attributes, C: Number of classes, N: Sample size

Dataset	d	C	N
balance	4	3	625
cmc	9	3	1473
hayesroth	4	3	160
iris	4	3	150
leukemia1	5327	3	72
leukemia2	11225	3	72
splice	60	3	3175
tae	5	3	151
wave	21	3	5000
wine	13	3	178
braintumor2	10367	4	50
car	6	4	1728
srbct	2308	4	83
vehicle	18	4	846
braintumor1	5920	5	90
pageblock	10	5	5473
dermatology	34	6	366
segment	19	7	2310
led7	7	10	3200
mfeatmor	6	10	2000
ocr	256	10	600
yeast	8	10	1484

- Rnd: We randomly choose K base classifiers with K different class orderings.
- Ren$_m$: The proposed classifier ensemble which uses m base classifiers in the beginning of the floating search.
- All: All the available base classifiers are combined without selection. For $K \leq 6$, we run classifiers for all possible class orderings. For $K > 6$, we combine all classifiers generated in both versions of Ren$_1$ and Ren$_K$.

V. Experimental Results

We can see the average error rates of the algorithms in Table II. The proposed Ren$_K$ algorithm is almost always more accurate than the compared algorithms. Rnd and Best are interchangebly better than each other (11 Best, 10 Rnd and a tie) but they are worse than Ren$_K$ which confirms that choosing a single classifier or randomly creating an ensemble does not lead to better classification accuracy.

Average number of base classifiers chosen by the variants of the Ren$_1$ and Ren$_K$ algorithms are shown in Table III. We can see that Ren algorithms choose significantly small number of classifiers with better accuracies than All and Best. Using this approach, we can have better accuracy with training a small number of classifiers and combining a subset of them.

Another interesting result is that Ren$_K$ accuracy is better than or equal to Ren$_1$ except for two cases (hayesroth and car). This shows us that it is not always better to start the floating search from the best classifier. This is mainly because since Best is already a good classifier, it usually dominates any classifier combined with it and the combination accuracy does not increase. Starting the floating search process using K classifiers introduces diversity into the ensemble from the start and leads to better ensemble accuracies.

TABLE II
Average error rates of the algorithms

Dataset	Best	Rnd	All	Ren$_K$	Ren$_1$
balance	27.837	28.192	26.106	25.855	26.169
cmc	47.196	47.596	46.870	46.541	46.451
hayesroth	32.184	33.742	30.120	29.374	29.145
iris	5.441	6.900	5.650	5.585	6.130
leukemia1	21.812	23.460	22.572	19.895	20.152
leukemia2	49.819	45.854	44.167	44.167	44.167
splice	6.456	6.498	5.435	5.378	5.378
tae	66.265	66.387	66.469	66.265	66.306
wave	24.400	20.729	18.944	18.950	18.950
wine	8.989	10.298	5.259	4.928	5.945
braintumor2	71.961	71.961	71.961	71.961	71.961
car	9.783	11.976	10.565	8.625	8.512
srbct	34.725	30.398	24.451	24.865	26.514
vehicle	36.441	35.476	32.775	32.162	32.514
braintumor1	31.034	31.241	31.034	30.793	31.103
pageblock	3.565	3.430	3.236	3.118	3.256
dermatology	3.404	4.534	3.405	3.364	5.156
segment	6.286	5.300	4.104	3.940	4.831
led7	29.389	28.017	27.240	26.965	29.179
mfeatmor	30.436	28.385	27.794	27.586	28.408
ocr	32.400	22.270	18.100	18.210	23.660
yeast	43.390	42.542	41.706	41.158	42.297

TABLE III
Average ensemble size of Ren$_K$, Ren$_1$, and All

Dataset	Ren$_K$	Ren$_1$	All
balance	3.4	3.0	6
cmc	2.9	2.0	6
hayesroth	3.6	3.1	6
iris	3.0	2.0	6
leukemia1	4.6	4.2	6
leukemia2	4.1	4.3	6
splice	5.0	5.0	6
tae	2.5	1.4	6
wave	6.0	6.0	6
wine	3.8	3.2	6
braintumor2	4.0	1.0	24
car	4.1	2.7	24
srbct	6.6	5.1	24
vehicle	6.8	7.4	24
braintumor1	5.7	1.7	120
pageblock	8.2	6.3	120
dermatology	9.2	2.9	720
segment	14.9	10.9	1161
led7	18.2	5.0	2075
mfeatmor	14.7	10.2	2151
ocr	18.2	13.5	2542
yeast	15.7	6.4	1983

Table IV shows average number of states visited by Ren$_K$ and Ren$_1$ algorithms in the floating search. Nearly in all datasets, Ren$_1$ visits less states than Ren$_K$, and therefore requires less training time (and combination time) compared to Ren$_K$. Given the number of possible states is $2^{K!}$, these results show that with traversing a very tiny part of the ensemble search space, we can get both significantly smaller and significantly better ensembles compared to ensembles such as All.

TABLE IV
AVERAGE NUMBER OF STATES VISITED BY REN$_K$ AND REN$_1$

Dataset	REN$_K$	REN$_1$
balance	14.0	11.7
cmc	16.6	14.8
hayesroth	14.4	14.3
iris	13.0	7.4
leukemia1	16.4	16.5
leukemia2	11.8	15.3
splice	18.2	18.8
tae	8.7	4.2
wave	20.0	22.7
wine	13.9	12.0
braintumor2	14.5	4.0
car	91.3	43.3
srbct	66.0	45.5
vehicle	91.6	82.3
braintumor1	41.4	9.9
pageblock	149.2	101.1
dermatology	204.0	35.4
segment	701.3	328.4
led7	1614.2	168.3
mfeatmor	786.1	607.6
ocr	1333.2	866.8
yeast	1003.4	270.0

VI. DISCUSSION

In this work, we propose a novel classifier combination scheme based on rule learners where the base classifiers are trained on the run (are not already trained like the methods discussed above) and there are more base classifiers to choose from. To our knowledge, there is no work in the literature addressing the combination of this many classifiers and our REN$_K$ approach aims to solve this problem using rule learners. We show that REN$_K$ algorithm almost always outperforms the other methods and with a small number of classifiers in the ensemble (compared to $K!$ and ALL).

There exists several methods for combination. The easiest method is voting which corresponds to the sum rule (averaging) or other fixed rules (i.e. min, max, product, median) [13]. There are methods which sample the data set (bagging [14], AdaBoost [15] and random subspace method [16]) but they are not used with different classification algorithms. There are methods which use another layer of classifier to fuse the outputs of the base classifiers [17]. In a mixture of experts architecture (MoE), classifiers are local and a separate gating network selects one of the local experts based on the input [18].

Unless the number of base classifiers is small [19], it is not possible to exhaustively combine all base classifiers; therefore methods have been proposed to choose a small subset from a large set of candidate classifiers or decorrelate base classifiers to reduce "diversity" [20]. A special journal issue edited by Kuncheva [21] contains papers on diversity measures and their role in ensemble construction. Although most studies show superiority of direct combiner error based search used as selection criteria in favor of the diversity measures [22], [23] or show that the the relation between accuracy and diversity is unclear [24], [25], there are studies [26] where compound diversity measures, which combine accuracy and diversity, are proposed. Methods preferred for the purpose of pruning ensembles: [22], [23], [27], [28]) use forward search, ([29], [30]) use backward search, ([22], [29], [31]) use genetic algorithms, [32], [33] use meta-learning and [34], [35] use eigenclassifiers. Greedy heuristic algorithms, such as forward and backward search have been shown to find ensembles having accuracies as high as the optimal subset constructed by exhaustive search [22], [23]. Forward subset selection has been shown to outperform classical combination schemes such as Bagging and boosting [27].

Several studies have been conduct for the purpose of comparing subset selection algorithms ([22], [23], [27]). Of the search strategies, forward and backward searches find ensembles which have the same accuracy, but forward search is faster, finds simpler ensembles because backward search may stop early especially when the number of base classifiers is high [22], [23]. When there are a few base classifiers, floating search does not add to the accuracy of forward search, because most of the time the chosen subsets have a small number of base classifiers, and pruning a previously added classifier is not necessary [23]. More similar to our case, Cabrera [36] investigates average, median and maximum rules when the number of classifiers becomes large. He finds that for normal error average is the best, for uniform error maximum is the best and for Cauchy error median is the best. Dos Santos et al. [4] propose a dynamic overproduce and choose strategy where they use random subspaces and bagging for overproduction phase and use genetic algorithms for the selection phase.

REFERENCES

[1] L. I. Kuncheva, *Combining pattern classifiers: methods and algorithms*. Wiley-Interscience, 2004.

[2] E. Alpaydın, *Introduction to Machine Learning*. The MIT Press, 2010.

[3] Y. Zhang, S. Burer, and W. N. Street, "Ensemble pruning via semi-definite programming," *Journal of Machine Learning Research*, vol. 7, pp. 1315–1338, 2006.

[4] E. M. D. Santos, R. Sabourin, and P. Maupin, "A dynamic overproduce-and-choose strategy for the selection of classifier ensembles," *Pattern Recognition*, vol. 41, pp. 2993–3009, 2008.

[5] J. Fürnkranz, "Separate-and-conquer learning," *Artificial Intelligence Review*, vol. 13, pp. 3–54, 1999.

[6] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Meteo, CA: Morgan Kaufmann, 1993.

[7] E. Frank and I. H. Witten, "Generating accurate rule sets without global optimization," in *Proceedings of the 15th International Conference on Machine Learning*, 1998, pp. 144–151.

[8] P. Clark and R. Boswell, "Rule induction with CN2: Some recent improvements," in *Lecture Notes in Artificial Intelligence*, vol. 482, 1990, pp. 151–163.

[9] W. W. Cohen, "Fast effective rule induction," in *The Twelfth International Conference on Machine Learning*, 1995, pp. 115–123.

[10] C. Blake and C. Merz, "UCI repository of machine learning databases," 2000. [Online]. Available: http://www.ics.uci.edu/∼mlearn/MLRepository.html

[11] A. Statnikov, C. Aliferis, I. Tsamardinos, D. Hardin, and S. Levy, "A comprehensive evaluation of multicategory classification methods for microarray gene expression cancer diagnosis," *Bioinformatics*, vol. 21, pp. 631–643, 2005.

[12] T. G. Dietterich, "Approximate statistical tests for comparing supervised classification learning classifiers," *Neural Computation*, vol. 10, pp. 1895–1923, 1998.

[13] J. Kittler, M. Hatef, R. P. Duin, and J. Matas, “On combining classifiers,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 3, pp. 226–239, 1998.

[14] L. Breiman, “Bagging predictors,” *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[15] Y. Freund and R. E. Schapire, “Experiments with a new boosting algorithm,” in *Proceedings of the International Conference on Machine Learning, ICML ’96*, 1996, pp. 148–156.

[16] T. K. Ho, “The random subspace method for constructing decision forests,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 8, pp. 832–844, 1998.

[17] D. H. Wolpert, “Stacked generalization,” *Neural Networks*, vol. 5, pp. 241–259, 1992.

[18] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, “Adaptive mixtures of local experts,” *Neural Computation*, vol. 3, pp. 79–87, 1991.

[19] A. J. C. Sharkey, N. E. Sharkey, U. Gerecke, and G. O. Chandroth, “The “test and select” approach to ensemble combination,” in *Proceedings of the International Workshop on Multiple Classifier Systems, MCS ’00*, vol. 1857, 2000, pp. 30–44.

[20] C. Yang, X.-C. Yin, and H.-W. Hao, “Diversity-based ensemble with sample weight learning,” in *Pattern Recognition (ICPR), 2014 22nd International Conference on*, Aug 2014, pp. 1236–1241.

[21] L. I. Kuncheva, “Special issue on diversity in multiple classifier systems.” *Information Fusion*, vol. 6, no. 1, pp. 1–115, 2005.

[22] D. Ruta and B. Gabrys, “Classifier selection for majority voting,” *Information Fusion*, vol. 6, no. 1, pp. 63–81, 2005.

[23] A. Ulaş, M. Semerci, O. T. Yıldız, and E. Alpaydın, “Incremental construction of classifier and discriminant ensembles,” *Information Sciences*, vol. 179, no. 9, pp. 1298–1318, April 2009.

[24] L. I. Kuncheva, M. Skurichina, and R. P. W. Duin, “An experimental study on diversity for bagging and boosting with linear classifiers,” *Information Fusion*, vol. 3, no. 4, pp. 245–258, 2002.

[25] L. I. Kuncheva and C. J. Whitaker, “Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy,” *Machine Learning*, vol. 51, no. 2, pp. 181–207, 2003.

[26] A. H.-R. Ko, R. Sabourin, and A. de Souza Britto Jr., “A new objective function for ensemble selection in random subspaces,” in *Proceedings of the International Conference on Pattern Recognition (ICPR ’06)*, 2006, pp. 185–188.

[27] R. Caruana, A. Niculescu-Mizil, G. Crew, and A. Ksikes, “Ensemble selection from libraries of models,” in *Proceedings of the International Conference on Machine Learning, ICML ’04*, 2004, pp. 137–144.

[28] Y. Yang, G. I. Webb, J. Cerquides, K. B. Korb, J. Boughton, and K. M. Ting, “To select or to weigh: A comparative study of linear combination schemes for superparent-one-dependence estimators,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 12, pp. 1652–1665, 2007.

[29] D. Partridge and W. B. Yates, “Engineering multiversion neural-net systems,” *Neural Computation*, vol. 8, no. 4, pp. 869–893, 1996.

[30] D. D. Margineantu and T. G. Dietterich, “Pruning adaptive boosting,” in *Proceedings of the International Conference on Machine Learning, ICML ’97*, 1997, pp. 211–218.

[31] Z.-H. Zhou, J. Wu, and W. Tang, “Ensembling neural networks: many could be better than all,” *Artificial Intelligence*, vol. 137, pp. 239–263, 2002.

[32] J. H. Krijthe, T. K. Ho, and M. Loog, “Improving cross-validation based classifier selection using meta-learning,” in *Pattern Recognition (ICPR), 2012 21st International Conference on*, Nov 2012, pp. 2873–2876.

[33] R. M. O. Cruz, R. Sabourin, and G. D. C. Cavalcanti, “On meta-learning for dynamic ensemble selection,” in *Pattern Recognition (ICPR), 2014 22nd International Conference on*, Aug 2014, pp. 1230–1235.

[34] A. Ulaş, O. T. Yıldız, and E. Alpaydın, “Eigenclassifiers for combining correlated classifiers,” *Information Sciences*, vol. 187, pp. 109–120, March 2012.

[35] Ümit Ekmekçi and Z. Çataltepe, “Extended multimodal eigenclassifiers and criteria for fusion model selection,” *Information Sciences*, vol. 298, pp. 53–65, March 2015.

[36] J. B. D. Cabrera, “On the impact of fusion strategies on classification errors for large ensembles of classifiers,” *Pattern Recognition*, vol. 39, pp. 1963–1978, 2006.