Focused Ensemble Selection: A Diversity-Based Method for Greedy Ensemble Selection

Ioannis Partalas, Grigorios Tsoumakas, Ioannis Vlahavas[1]

Abstract. Ensemble selection deals with the reduction of an ensemble of predictive models in order to improve its efficiency and predictive performance. A number of ensemble selection methods that are based on greedy search of the space of all possible ensemble subsets have recently been proposed. This paper contributes a novel method, based on a new diversity measure that takes into account the strength of the decision of the current ensemble. Experimental comparison of the proposed method, dubbed Focused Ensemble Selection (FES), against state-of-the-art greedy ensemble selection methods shows that it leads to small ensembles with high predictive performance.

1 Introduction

Ensemble methods [6] has been a very popular research topic during the last decade. Their success arises from the fact that they offer an appealing solution to several interesting learning problems of the past and the present, such as: improving predictive performance over a single model, scaling inductive algorithms to large databases, learning from multiple physically distributed data sets and learning from concept-drifting data streams.

Typically, ensemble methods comprise two phases: the *production* of multiple predictive models and their *combination*. Recent work [9, 8, 7, 15, 4, 10, 11, 2], has considered an additional intermediate phase that deals with the reduction of the ensemble size prior to combination. This phase is commonly named *ensemble pruning*, *selective ensemble*, *ensemble thinning* and *ensemble selection*, the last one of which is used within this paper.

Ensemble selection is important for two reasons: *efficiency* and *predictive performance*. Having a very large number of models in an ensemble adds a lot of computational overhead. For example, decision tree models may have large memory requirements [9] and lazy learning methods have a considerable computational cost during execution. The minimization of run-time overhead is crucial in certain applications, such as stream mining. Equally important is the second reason, predictive performance. An ensemble may consist not only of high performance models, but also of models with lower predictive performance. Intuitively, combining good and bad models together will not have the expected result. Pruning the low-performing models while maintaining a good diversity of the ensemble is typically considered as a proper recipe for a successful ensemble.

The problem of pruning an ensemble of classifiers has been proved to be NP-complete [14]. Exhaustive search for the best subset of classifiers isn't tractable for ensembles that contain a large number of models. Greedy approaches, such as [2, 4, 9, 10, 11], are fast, as they consider a very small part of the space of all combinations. These methods, start with an initial ensemble (empty or full) and search in the space of the different ensembles, by iteratively expanding or contracting the initial ensemble by a single model. The search is guided by either the predictive performance or the diversity of the alternative ensembles.

This paper contributes a novel method for greedy ensemble selection, based on a new diversity measure that takes into account the strength of the decision of the current ensemble. Experimental comparison of the proposed method, dubbed Focused Ensemble Selection (FES), against state-of-the-art greedy ensemble selection methods shows that it leads to small ensembles with high predictive performance.

The remainder of this paper is structured as follows: Section 2 presents background information on ensemble methods and Section 3 reviews previous work on ensemble selection. Section 4 introduces the proposed method. Section 5 presents the setup of the experimental study and Section 6 discusses the results. Finally, Section 7 concludes this work.

2 Ensemble Methods

2.1 Producing the Models

An ensemble can be composed of either homogeneous or heterogeneous models. Homogeneous models derive from different executions of the same learning algorithm by using different values for the parameters of the learning algorithm, injecting randomness into the learning algorithm or through the manipulation of the training instances, the input attributes and the model outputs [6]. Two popular methods for producing homogeneous models are bagging [3] and boosting [13].

Heterogeneous models derive from running different learning algorithms on the same dataset. Such models have different views about the data, as they make different assumptions about them. For example, a neural network is robust to noise in contrast to a k-nearest neighbor classifier.

2.2 Combining the Models

A lot of different ideas and methods have been proposed in the past for the combination of classification models. The main motivation underlying this research is the observation that there is no single classifier that performs significantly better in every classification problem [18]. The necessity for high classification performance in some critical domains (e.g. medical, financial, intrusion detection) have

[1] Department of Informatics, Aristotle University of Thessaloniki, 54124 Thessaloniki, Greece, email: {partalas,greg,vlahavas}@csd.auth.gr

urged researchers to explore methods that combine different classification algorithms in order to overcome the limitations of individual learning paradigms.

Unweighted and Weighted Voting are two of the simplest methods for combining not only Heterogeneous but also Homogeneous models. In Voting, each model outputs a class value (or ranking, or probability distribution) and the class with the most votes (or the highest average ranking, or average probability) is the one proposed by the ensemble. In Weighted Voting, the classification models are not treated equally. Each model is associated with a coefficient (weight), usually proportional to its classification accuracy.

Let x be an instance and m_i, $i = 1..k$ a set of models that output a probability distribution $m_i(x, c_j)$ for each class c_j, $j = 1..n$. The output of the (weighted) voting method $y(x)$ for instance x is given by the following mathematical expression:

$$y(x) = \arg\max_{c_j} \sum_{i=1}^{k} w_i m_i(x, c_j),$$

where w_i is the weight of model i. In the simple case of voting (unweighted), the weights are all equal to one, that is, $w_i = 1, i = 1..k$.

3 Ensemble Selection

3.1 Greedy Approaches

Margineantu and Dietterich [9] introduce heuristics to calculate the benefit of adding a classifier to an ensemble, using forward selection in a number of them. These heuristics are based on the diversity and the performance of the classifiers. The authors experiment with boosting ensembles and conclude that pruning can help an ensemble to increase its predictive performance.

Fan et al. [7] prune an ensemble of classifiers using forward selection of the classification models, like in [9]. As a heuristic, they use the benefit that is obtained by evaluating the combination of the selected classifiers with the method of voting. Their results show that pruning increases the predictive performance and speeds up the run time of an ensemble of C4.5 decision trees trained on disjoint parts of a large data set.

Caruana et al. [4] produce an ensemble of 1000 classifiers using different algorithms and sets of parameters for these algorithms. They subsequently prune the ensemble following an approach that is similar to [9]. This way they manage to achieve very good predictive performance compared to state-of-the-art ensemble methods.

Banfield et al. [2], propose a method that selects a subensemble in a backward manner. The authors reward each classifier according to its decision with regard to the ensemble decision. The method removes the classifier with the lowest accumulated reward.

Martinez-Munoz et al. [11, 10] present two algorithms for pruning an ensemble of classifiers. In [11] the authors define for each classifier a vector with dimensionality equal to the size of the training set, where each element i corresponds to the decision of the classifier for the instance i. The classifier is added to the ensemble according to its impact in the difference between the vector of the ensemble (average of individual vectors) with a predefined reference vector. This reference vector indicates the desired direction towards which the vector of the ensemble must align. In [10], the authors produce an initial ensemble of bagging models. Then using a forward selection procedure, they add to the ensemble the classifier that disagrees the most with the current ensemble. The process ends when a predefined size for the final pruned ensemble is reached.

3.2 Other Approaches

Giacinto and Roli [8] employ Hierarchical Agglomerative Clustering (HAC) for ensemble selection. This way they implicitly used the *complete link* method for inter-cluster distance computation. Pruning is accomplished by selecting a single representative classifier from each cluster. The representative classifier is the one exhibiting the maximum average distance from all other clusters.

Zhou and Tang [20] perform stochastic search in the space of model subsets using a standard genetic algorithm. Standard genetic operations such as mutations and crossovers are used and default values are used for the parameters of the genetic algorithm. The voted performance of the ensemble is used as a function for evaluating the fitness of individuals in the population.

Tsoumakas et al. [15] prune an ensemble of heterogeneous classifiers using statistical procedures that determine whether the differences in predictive performance among the classifiers of the ensemble are significant. Only the classifiers with significantly better performance than the rest are retained and subsequently combined with the methods of (weighted) voting.

Zhang et al. [19], formulate the ensemble pruning problem as a mathematical problem and apply semi-definite programming (SDP) techniques. Their algorithm requires the number of classifiers to retain as a parameter and runs in polynomial time.

Partalas et al. [12], present an ensemble selection method under the framework of Reinforcement Learning, where the learning module finds an optimal policy for including or excluding a classifier from the ensemble.

4 Focused Ensemble Selection

Let $H = \{h_t, t = 1, 2, \ldots, T\}$ be the set of classifiers (or hypotheses) of an ensemble, where each classifier h_t maps an instance $\boldsymbol{x}$ to a class label y, $h_t(\boldsymbol{x}) = y$. Greedy ensemble selection approaches start either with an empty set of classifiers ($S = \emptyset$) or the complete ensemble ($S = H$). For simplicity of presentation we focus on the former initial conditions only, yet our argumentation holds for both.

At each step the current subset S is expanded by a model $h_t \in H \setminus S$, based on either the predictive performance [9, 7, 4] or the diversity [9, 11, 10, 2] of the expanded ensemble $S \bigcup \{h_t\}$. Methods that are based on diversity have been shown to be more effective than those that are based on accuracy. The methods in [10, 2] measure the diversity of candidate ensembles $S \bigcup \{h_t\}$ by comparing the decision of the current ensemble S with the decision of candidate classifiers $h_t \in H \setminus S$ on a set of evaluation examples $(\boldsymbol{x_i}, y_i), i = 1, 2, \ldots, N$. Each example consists of a feature vector $\boldsymbol{x_i}$ and a class label y_i. We can distinguish 4 events concerning both of these decisions:

$$\begin{aligned} e_{tf} &: y = h_t(\boldsymbol{x_i}) \wedge y \neq S(\boldsymbol{x_i}) \\ e_{ft} &: y \neq h_t(\boldsymbol{x_i}) \wedge y = S(\boldsymbol{x_i}) \\ e_{tt} &: y = h_t(\boldsymbol{x_i}) \wedge y = S(\boldsymbol{x_i}) \\ e_{ff} &: y \neq h_t(\boldsymbol{x_i}) \wedge y \neq S(\boldsymbol{x_i}) \end{aligned}$$

where $S(\boldsymbol{x_i})$ is the classification of instance $\boldsymbol{x_i}$ by ensemble S. This classification is derived from the application of an ensemble combination method on S, which usually is voting.

The diversity measure in [10] is based on e_{tf} only, while the one in [2] neglects e_{ft}. We argue that all events should contribute to the calculation of an appropriate diversity measure. Event e_{ft} for example, corresponds to the case where the candidate classifier errs, while

the ensemble is correct. Although, the ensemble is correct, we do not know how many votes lead to its correct decision. If the difference in votes between the correct and wrong decision is marginal, then this candidate classifier might lead to a misclassification of example x_i by the ensemble $S \bigcup \{h_t\}$.

The above example concerning event e_{ft}, brings up another disadvantage of the methods in [10, 2]. The decisions of individual models within the current ensemble are not separately considered, as the current ensemble is treated as a whole. We hypothesize that better results can be obtained from a measure that takes into account the *strength* of the current ensemble's decision. We argue that an example that is incorrectly (correctly) classified by most of the members of the current ensemble, should not affect strongly the ensemble selection method, as this is probably a very hard (easy) example. On the other hand, examples that are misclassified by about half of the ensemble's members, are near to change status (correct/incorrect classification) and should strongly affect the method.

In order to deal with the above issues, we propose a diversity measure that considers all events and takes into account the strength of the current ensemble's decision. We define the following quantities: NT_i, which denotes the proportion of models in the current ensemble S that classify example $(\boldsymbol{x_i}, y_i)$ correctly, and $NF_i = 1 - NT_i$, which denotes the number of models in S that classify it incorrectly.

The proposed method, dubbed Focused Ensemble Selection (FES), starts with the full ensemble ($S = H$) and iteratively removes the classifier $h_t \in S$ that minimizes the following quantity:

$$fes(h_t) = \sum_{i=1}^{N} \Big(NT_i * I(e_{tf}) - NF_i * I(e_{ft}) + \\ + NF_i * I(e_{tt}) - NT_i * I(e_{ff}) \Big),$$

where $I(true) = 1$ and $I(false) = 0$.

Note that events e_{tf} and e_{tt} increase the metric, because the candidate classifier is correct, while events e_{ft} and e_{ff} decrease it, as the candidate classifier is incorrect. The strength of increase/decrease depends on the strength of the ensemble's decision. If the current ensemble S is incorrect, then the reward/penalty is multiplied by the proportion of correct models in S. On the other hand, if S is correct, then the reward/penalty is multiplied by the proportion of incorrect models in S. This weighting scheme focuses the attention of the algorithm to examples that are near to change status, while it overlooks examples whose correct classification is either very easy or very hard.

In event e_{tf} for example, the addition of a correct classifier when the ensemble is wrong contributes a gain of 1 multiplied by the proportion of classifiers in that ensemble that are correct. The rationale is that if the number of classifiers is small, then correct classification of this example is hard to achieve and thus the contribution is penalized, while if the number of classifiers is large, then the correct classification of this example is easier to achieve and thus the contribution is rewarded.

An issue that is worth mentioning here concerns the dataset used for calculating the diversity (or predictive performance) measures in greedy ensemble selection methods. One approach is to use the training set for evaluation, as in [11]. This offers the benefit that plenty of data will be available for evaluation and training, but is susceptible to overfitting. Another approach is to withhold a part of the training set for evaluation, as in [4, 2] and the REPwB method in [9]. This is less prone to overfitting, but reduces the amount of data that are available for training and evaluation compared to the previous approach. FES supports both of these approaches.

Another important issue that concerns ensemble selection methods, is when to stop adding classifiers in the ensemble, or, in other words, how many models should the final ensemble include. One solution is to perform the search until all models have been added into (removed from) the ensemble and select the ensemble with the highest accuracy on the evaluation set. This approach has been used in [4]. Others prefer to select a predefined number of models, expressed as a percentage of the original ensemble [9, 7, 11, 2]. FES supports both of these approaches, but follows the former by default, because it is more flexible and automated, since it doesn't require the specification of a percentage.

Algorithm 1 presents the proposed method in pseudocode. Its time complexity is $O(T^2|S|N)$, which can be optimized to $O(T^2N)$ if the predictions of the current ensemble are updated incrementally each time a classifier is removed from it.

Algorithm 1 The proposed method in pseudocode

Require: Ensemble of classifiers H
1: $S = H$
2: $B = \emptyset$
3: $acc = 0$
4: **while** $S \neq \emptyset$ **do**
5: $\quad h = \arg\min_{h_t \in S} fes(h_t)$
6: $\quad S = S \setminus \{h\}$
7: $\quad acc_{temp} = Accuracy(S)$
8: $\quad$ **if** $acc_{temp} > acc$ **then**
9: $\quad\quad acc = acc_{temp}$
10: $\quad\quad B = S$
11: $\quad$ **end if**
12: **end while**
13: **return** B

5 Experimental Setup

5.1 Datasets

We experimented on 12 data sets from the UCI Machine Learning repository [1]. Table 1 presents the details of these data sets (Folder in UCI server, number of instances, classes, continuous and discrete attributes, percentage of missing values). We avoided using datasets with less than 650 examples, so that an adequate amount of data is available for training, evaluation and testing.

Table 1. Details of data sets: Folder in UCI server, number of instances, classes, continuous and discrete attributes, percentage of missing values

id	UCI Folder	Inst	Cls	Cnt	Dsc	MV(%)
d1	car	1728	4	0	6	0.00
d2	cmc	1473	3	2	7	0.00
d3	credit-g	1000	2	7	13	0.00
d4	kr-vs-kp	3196	2	0	36	0.00
d5	hypothyroid	3772	4	7	23	5.40
d6	segment	2310	7	19	0	0.00
d7	sick	3772	2	7	23	5.40
d8	soybean	683	19	0	35	0.00
d9	tic-tac-toe	958	2	0	9	0.00
d10	vehicle	946	4	18	0	0.00
d11	vowel	990	11	3	10	0.00
d12	waveform-5000	5000	3	21	0	0.00

5.2 Methodology

The methodology of the experiments proceeds as follows: Initially, the whole dataset is split into three disjunctive parts, a training set, an evaluation set and a test set with 40%, 40% and 20% of the initial dataset respectively.

In this paper, we focus on ensembles of heterogeneous models. We therefore run different learning algorithms with different parameters on the training set, in order to produce 200 models that constitute the initial ensemble. The WEKA machine learning library [17] was used as the source of learning algorithms. We trained 24 multilayer perceptrons (MLPs), 60 kNNs, 110 support vector machines (SVMs), 2 naive Bayes classifiers and 4 decision trees. The different parameters used to train the algorithms were the following (the rest of the parameters were left unchanged in their default values):

- MLPs: we used 6 values for the nodes in the hidden layer {1, 2, 4, 8, 32, 128} and 4 values for the momentum term {0.0, 0.2, 0.5, 0.8}.
- kNNs: we used 20 values for k distributed evenly between 1 and the plurality of the training instances. We also used 3 weighting methods: no-weighting, inverse-weighting and similarity-weighting.
- SVMs: we used 11 values for the complexity parameter {10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 0.1, 1, 10, 100, 1000}, and 10 different kernels. We used 2 polynomial kernels (of degree 2 and 3) and 8 radial kernels (gamma $\in$ {0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 2}).
- Naive Bayes: we built one model with default parameters and one with kernel estimation.
- Decision trees: we used 2 values for the confidence factor ({0.25, 0.5}), and 2 values for Laplace smoothing ({true, false}).

We compare the performance of our approach, Focused Ensemble Selection (FES), against the following greedy ensemble selection methods: Forward Selection (FS) [4], Complementariness (COM) [10], Margin Distance Minimization (MDM) [11] and Concurrency Thining (CT) [2].

The evaluation set is used for the calculation of diversity and performance measure for all competing algorithms, because preliminary experiments have shown that it leads to significantly better results than using the training set in ensembles of heterogeneous models. Voting was used for model combination in FES, FS, COM and CT. Similarly to FES, all rival algorithms follow the approach of [4], which selects the ensemble with the highest accuracy on the evaluation set, instead of using an arbitrary percentage of selection. In addition, the following section discuses comparative results with alternative versions of the algorithms that select a fixed percentage (20%) of models. The resulting ensemble is evaluated on the test set, using voting for model combination.

We also calculate the performance of the best single model (BSM) in the ensemble, and the performance of the complete ensemble of 200 models (ALL), using voting for model combination, based on the performance of the models on the evaluation dataset. The whole experiment is performed 10 times for each dataset and the results are averaged.

6 Results and Discussion

Table 2 presents the classification accuracy of each algorithm on each dataset. The accuracy of the winning algorithm at each dataset is highlighted with bold typeface. A first observation is that the proposed approach achieves the best performance in most of the datasets (6), followed by BSM (3), CT (2), MDM and FS (1) and finally COM and ALL (0).

Table 2. Classification accuracy for each algorithm on each dataset.

id	FES	FS	COM	CT	MDM	BSM	ALL
d1	98.3	98.1	98.2	98.4	97.4	**99.4**	82.7
d2	**52.7**	52.4	51.2	52.3	51.5	42.8	47.6
d3	**74.4**	74.2	73.2	74.0	73.4	69.5	70.8
d4	99.0	**99.1**	99.0	99.0	97.9	95.4	95.6
d5	**99.3**	99.2	99.2	**99.3**	97.8	90.7	91.9
d6	96.9	96.9	96.9	96.8	96.5	**98.5**	97.8
d7	98.1	98.0	98.0	**98.2**	97.4	95.2	95.4
d8	91.5	91.1	91.0	91.6	**91.7**	90.1	89.8
d9	**98.7**	98.5	98.6	98.6	98.4	95.8	63.9
d10	**81.1**	80.1	80.8	80.9	79.1	64.4	75.3
d11	90.3	90.5	89.8	90.3	87.8	**98.9**	90.7
d12	**86.0**	85.7	85.7	85.9	84.4	72.7	80.7

According to [5], the appropriate way to compare two or more algorithms on multiple datasets is based on their average rank across all datasets. On each dataset, the algorithm with the highest accuracy gets rank 1.0, the one with the second highest accuracy gets rank 2.0 and so on. In case two or more algorithms tie, they all receive the average of the ranks that correspond to them.

Table 3 presents the rank of each algorithm on each dataset, along with the average ranks. The proposed approach has the best average rank (2.17), followed by CT (2.71), FS (3.29), COMP (4.0), MDM (4.92), BSM (5.33) and ALL (5.58). Although the difference of the average ranks between the 2^{nd} best algorithm (CT) and FES is small, CT achieves the highest accuracy (and rank) in only two datasets. We therefore argue that FES should be preferred over CT and the rest of its rivals for ensemble selection.

Table 3. Corresponding rank for each algorithm on each dataset.

id	FES	FS	COM	CT	MDM	BSM	ALL
d1	3.0	5.0	4.0	2.0	6.0	1.0	7.0
d2	1.0	2.0	5.0	3.0	4.0	7.0	6.0
d3	1.0	2.0	5.0	3.0	4.0	7.0	6.0
d4	3.0	1.0	3.0	3.0	5.0	7.0	6.0
d5	1.5	3.5	3.5	1.5	5.0	7.0	6.0
d6	4.0	4.0	4.0	6.0	7.0	1.0	2.0
d7	2.0	3.5	3.5	1.0	5.0	7.0	6.0
d8	3.0	4.0	5.0	2.0	1.0	6.0	7.0
d9	1.0	4.0	2.5	2.5	5.0	6.0	7.0
d10	1.0	4.0	3.0	2.0	5.0	7.0	6.0
d11	4.5	3.0	6.0	4.5	7.0	1.0	2.0
d12	1.0	3.5	3.5	2.0	5.0	7.0	6.0
Av. Rank	2.17	3.29	4.0	2.71	4.92	5.33	5.58

We next turn to statistical procedures, in order to investigate whether the performance differences between FES and the rest of the algorithms are significant. According to [5], the appropriate statistical test for the comparison of two algorithms on multiple datasets is the Wilcoxon signed rank test [16]. Note that the majority of past approaches have used the paired t-test, which is inappropriate for this task. We performed 6 tests, one for each paired comparison of FES with each of the other algorithms, at a confidence level of 95%. The test found that FES is significantly better than all other algorithms, apart from CT.

Table 4 shows the average size of the final ensembles that are selected by the algorithms on each dataset. A general remark is that the number of selected models is small compared to the size of the original ensemble. Only 5.05% to 14.95% of the 200 classifiers are finally selected by the algorithms. Furthermore, the number of models selected based on the maximum accuracy in the evaluation set, is smaller than using a fixed size, such as 20% [10, 11] or 10% [2] of the models, leading to further reduction of the computational cost of the final ensemble.

Table 4. Average size of selected ensembles for each algorithm.

id	FES	FS	COMP	CT	MDM
d1	11.6	6.1	5.1	6.5	20.9
d2	18.3	15.9	13.4	16.7	26.1
d3	15.7	14.4	20.9	12.7	33.7
d4	13.4	11.1	11.8	11.2	27.9
d5	11.7	5.8	5.0	7.2	6.7
d6	20.5	17.2	17.8	15.3	29.6
d7	7.2	3.9	3.5	3.7	9.3
d8	23.3	13.7	11.0	10.9	40.0
d9	27.9	9.4	11.1	11.2	31.9
d10	8.6	13.3	10.3	10.6	21.9
d11	8.5	8.3	6.8	4.4	31.8
d12	20.8	40.6	15.8	15.1	79.1
Av. Size	15.6	13.3	11.0	10.5	29.9

In order to investigate whether the performance of greedy ensemble selection algorithms is significantly better when the size of the final ensemble is selected dynamically, rather than using a predefined percentage of models (20%), we performed Wilcoxon tests on the predictive performance of the two alternative versions of each algorithm on all datasets. With 95% confidence the test showed no statistical differences, but the results were in favor of the dynamic approach.

Figure 1 presents the mean number of each type of models that are selected by FSD across all datasets for the type of models that are selected. FSD selects on average 7.5 SVMs, 5.2 MLPs, 1.3 kNNs, 1.2 DTs and 0.4 NB models. This shows that SVMs and MLPs, which are traditionally highly accurate classifiers, dominate the final ensembles. On the other hand we notice that the final ensembles include on average 30% of the trained DTs, 22% of the trained MLPs and NB models, 7% of SVMs and 2% of kNNs. This shows that our production procedure led to quite diverse DTs, MLPs and NB models, while on the other hand most of the produced SVMs and kNNs were probably very similar. These results can be taken into account, in order to produce a more diverse initial ensemble.

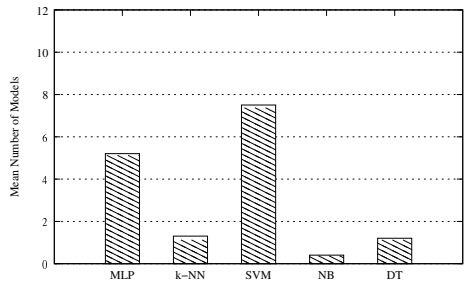

Figure 1. Aggregates for FSD concerning the type of models that are selected.

7 Conclusions

This paper contributed a new method for greedy ensemble selection, named Focused Ensemble Selection (FES). The main idea of the method is to overlook examples that are either very easy or very hard, and focus on those that are near to change status (correct/incorrect classification).

We performed experiments comparing FES with state-of-the-art methods from the related bibliography. Although FES was not found significantly better than all competitors that were considered in this paper, it still was found consistently better based on both the average rank and the number of datasets, where it achieved the highest accuracy. We consider that the main novel idea of this paper (taking into consideration the strength of the ensemble's classification) is a positive contribution that could be valuable to other researchers working in ensemble selection and ensemble methods in general.

REFERENCES

[1] A. Asuncion and D.J. Newman. UCI machine learning repository, 2007. http://www.ics.uci.edu/∼mlearn/MLRepository.html.

[2] Robert E. Banfield, Lawrence O. Hall, Kevin W. Bowyer, and W. Philip Kegelmeyer, 'Ensemble diversity measures and their application to thinning.', *Information Fusion*, **6**(1), 49–62, (2005).

[3] L. Breiman, 'Bagging Predictors', *Machine Learning*, **24**(2), 123–40, (1996).

[4] R. Caruana, A. Niculescu-Mizil, G. Crew, and A. Ksikes, 'Ensemble selection from libraries of models', in *Proceedings of the 21st International Conference on Machine Learning*, (2004).

[5] Janez Demsar, 'Statistical comparisons of classifiers over multiple data sets', *Journal of Machine Learning Research*, **7**, 1–30, (2006).

[6] T. G. Dietterich, 'Ensemble Methods in Machine Learning', in *Proceedings of the 1st International Workshop in Multiple Classifier Systems*, pp. 1–15, (2000).

[7] Wei Fan, Fang Chu, Haixun Wang, and Philip S. Yu, 'Pruning and dynamic scheduling of cost-sensitive ensembles', in *Eighteenth national conference on Artificial intelligence*, pp. 146–151. AAAI, (2002).

[8] Giorgio Giacinto and Fabio Roli, 'An approach to the automatic design of multiple classifier systems', *Pattern Recognition Letters*, **22**(1), 25–33, (2001).

[9] D. Margineantu and T. Dietterich, 'Pruning adaptive boosting', in *Proceedings of the 14th International Conference on Machine Learning*, pp. 211–218, (1997).

[10] G. Martinez-Munoz and A. Suarez, 'Aggregation ordering in bagging', in *International Conference on Artificial Intelligence and Applications (IASTED)*, pp. 258–263. Acta Press, (2004).

[11] G. Martinez-Munoz and A. Suarez, 'Pruning in ordered bagging ensembles', in *23rd International Conference in Machine Learning (ICML-2006)*, pp. 609–616. ACM Press, (2006).

[12] I. Partalas, G. Tsoumakas, I. Katakis, and I. Vlahavas, 'Ensemble pruning using reinforcement learning', in *4th Hellenic Conference on Artificial Intelligence (SETN 2006)*, pp. 301–310, (May 18–20 2006).

[13] Robert E. Schapire, 'The strength of weak learnability', *Machine Learning*, **5**, 197–227, (1990).

[14] Christino Tamon and Jie Xiang, 'On the boosting pruning problem', in *11th European Conference on Machine Learning (ECML 2000)*, pp. 404–412. Springer-Verlag, (2000).

[15] G. Tsoumakas, L. Angelis, and I. Vlahavas, 'Selective fusion of heterogeneous classifiers', *Intelligent Data Analysis*, **9**(6), 511–525, (2005).

[16] F. Wilcoxon, 'Individual comparisons by ranking methods', *Biometrics*, **1**, 80–83, (1945).

[17] I.H. Witten and E. Frank, *Data Mining: Practical machine learning tools and techniques*, Morgan Kaufmann, 2005.

[18] D. Wolpert, *The mathematics of generalization*, Addison-Wesley, 1995.

[19] Yi Zhang, Samuel Burer, and W. Nick Street, 'Ensemble pruning via semi-definite programming', *Journal of Machine Learning Research*, **7**, 1315–1338, (2006).

[20] Zhi-Hua Zhou and Wei Tang, 'Selective ensemble of decision trees', in *9th International Conference on Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing, RSFDGrC 2003*, pp. 476–483, Chongqing, China, (May 2003).