A Taxonomy and Short Review of Ensemble Selection

Grigorios Tsoumakas and **Ioannis Partalas** and **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. The last 10 years a large number of very diverse ensemble selection methods have been proposed. In this paper we make a first approach to categorize them into a taxonomy. We also present a short review of some of these methods. We particularly focus on a category of methods that are based on greedy search of the space of all possible ensemble subsets. Such methods use different directions for searching this space and different measures for evaluating the available actions at each state. Some use the training set for subset evaluation, while others a separate validation set. This paper abstracts the key points of these methods and offers a general framework of the greedy ensemble selection algorithm, discussing its important parameters and the different options for instantiating these parameters.

1 Introduction

Ensemble methods [5] have been a very popular research topic during the last decade. They have attracted scientists from several fields including Statistics, Machine Learning, Pattern Recognition and Knowledge Discovery in Databases. Their popularity arises largely from the fact that they offer an appealing solution to several interesting learning problems of the past and the present.

First of all, ensembles *lead to improved accuracy* compared to a single classification or regression mode. This was the main motivation that led to the development of the ensemble methods area. Ensembles achieve higher accuracy than individual models, mainly through the correction of their uncorrelated errors. Secondly, ensembles *solve the problem of scaling inductive algorithms to large databases*. Most inductive algorithms are too computationally complex and suffer from memory problems when applied to very large databases. A solution to this problem is to horizontally partition the database into smaller parts, train a predictive model in each of the smaller manageable part and combine the predictive models. Thirdly, ensembles can *learn from multiple physically distributed data sets*. Often such data can't be collected to a single site due to privacy or size reasons. This problem can be overcome through the combination of multiple predictive models, each trained on a different distributed data set. Finally, ensembles are useful for *learning from concept-drifting data streams*. The main idea here is to maintain an ensemble of classifiers that are trained from different batches of the data stream. Combining these classifiers with a proper methodology can solve the problem of data expiration that occurs when the learning concept drifts.

Typically, ensemble methods comprise two phases: the *production* of multiple predictive models and their *combination*. Recent work [13, 9, 12, 7, 21, 4, 14, 1, 16, 24, 17], has considered an additional intermediate phase that deals with the reduction of the ensemble size prior to combination. This phase is commonly called *ensemble pruning*, *selective ensemble*, *ensemble thinning* and *ensemble selection*, of which we shall use the last one 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 [13] and lazy learning methods have a considerable computational cost during execution. The minimization of run-time overhead is crucial in certain applications, such as in stream mining. Equally important is the second reason, predictive performance. An ensemble may consist of both high and low predictive performance models. The latter may negatively affect the overall performance of the ensemble. Pruning these models while maintaining a high diversity among the remaining members of the ensemble is typically considered a proper recipe for an effective ensemble.

The last 10 years a large number of very diverse ensemble selection methods have been proposed. In this paper we make a first approach to categorize them into a taxonomy. We hope that community feedback will help fine-tuning this taxonomy and shape it into a proper starting place for researchers designing new methods. In addition, we delve a little deeper into a specific category in this taxonomy: greedy search-based methods.

A number of ensemble selection methods that are based on a greedy search of the space of all possible ensemble subsets, have recently been proposed [13, 7, 4, 14, 1]. They use different directions for searching this space and different measures for evaluating the available actions at each state. Some use the training set for subset evaluation, while others a separate validation set. In this paper we attempt to highlight the salient parameters of greedy ensemble selection algorithms, offer a critical discussion of the different options for instantiating these parameters and mention the particular choices of existing approaches. The paper steers clear of a mere enumeration of particular approaches in the related literature, by generalizing their key aspects and providing comments, categorizations and complexity analysis wherever possible.

The remainder of this paper is structured as follows. Section 2 contains background material on ensemble production and combination. Section 3 presents the proposed taxonomy including a short account of methods in each category. The category of clustering-based methods is discussed at a greater detail, from a more critical point of view. Section 4 discusses extensively the category of greedy search-based ensemble selection algorithms. Finally Section 5 concludes this work.

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

2 Background

This section provides background material on ensemble methods. More specifically, information about the different ways of producing models are presented as well as different methods for combining the decisions of the models.

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. Such models can be produced 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]. Popular methods for producing homogeneous models are *bagging* [2] and *boosting* [18].

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

2.2 Combining the Models

Common methods for combining an ensemble of predictive models include *voting*, *stacked generalization* and *mixture of experts*.

In voting, each model outputs a class value (or ranking, or probability distribution) and the class with the most votes is the one proposed by the ensemble. When the class with the maximum number of votes is the winner, the rule is called *plurality voting* and when the class with more than half of the votes is the winner, the rule is called *majority voting*. A variant of voting is weighted voting where the models are not treated equally as each of them 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, $w_i = 1, i = 1..k$.

Stacked generalization [23], also known as *stacking* is a method that combines models by learning a meta-level (or level-1) model that predicts the correct class based on the decisions of the base level (or level-0) models. This model is induced on a set of meta-level training data that are typically produced by applying a procedure similar to k-fold cross validation on the training data. The outputs of the base-learners for each instance along with the true class of that instance form a meta-instance. A meta-classifier is then trained on the meta-instances. When a new instance appears for classification, the output of the all base-learners is first calculated and then propagated to the meta-classifier, which outputs the final result.

The mixture of experts architecture [10] is similar to the weighted voting method except that the weights are not constant over the input space. Instead there is a gating network which takes as input an instance and outputs the weights that will be used in the weighted voting method for that specific instance. Each expert makes a decision and the output is averaged as in the method of voting.

3 A Taxonomy of Ensemble Selection Algorithms

We propose the organization of the various ensemble selection methods into the following categories: a) Search-based, b) Clustering-based c) Ranking-based and d) Other.

3.1 Search Based Methods

The most direct approach for pruning an ensemble of predictive models is to perform a heuristic search in the space of the possible different model subsets, guided by some metric for the evaluation of each candidate subset. We further divide this category into two subcategories, based on the search paradigm: a) greedy search, and b) stochastic search. The former is among the most popular categories of ensemble pruning algorithms and is investigated at depth in Section 4. Stochastic search allows randomness in the selection of the next candidate subset and thus can avoid getting stuck in local optima.

3.1.1 Stochastic Search

Gasen-b [25] performs stochastic search in the space of model subsets using a standard genetic algorithm. The ensemble is represented as a bit string, using one bit for each model. Models are included or excluded from the ensemble depending on the value of the corresponding bit. Gasen-b performs standard genetic operations such as mutations and crossovers and uses default values for the parameters of the genetic algorithm. The performance of the ensemble is used as a function for evaluating the fitness of individuals in the population.

Partalas et al. [16] search the space of model subsets using a reinforcement learning approach. We categorize this approach into the stochastic search algorithms, as the exploration of the state space includes a (progressively reducing) stochastic element. The problem of pruning an ensemble of n classifiers has been transformed into the reinforcement learning task of letting an agent learn an optimal policy of taking n actions in order to include or exclude each classifier from the ensemble. The method uses the Q-learning [22] algorithm to approximate an optimal policy.

3.2 Clustering-based methods

The methods of this category comprise two stages. Firstly, they employ a clustering algorithm in order to discover groups of models that make similar predictions. Subsequently, each cluster is separately pruned in order to increase the overall diversity of the ensemble.

3.2.1 Giacinto et al., 2000

Giacinto et al. [9] employ Hierarchical Agglomerative Clustering (HAC) for classifier pruning. This type of clustering requires the definition of a distance metric between two data points (here classifiers). The authors defined this metric as the probability that the classifiers don't make coincident errors and estimate it from a validation set in order to avoid overfitting problems. The authors also defined the distance between two clusters as the maximum distance between two classifiers belonging to these clusters. 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.

HAC returns a hierarchy of different clustering results starting from as many clusters as the data points and ending at a single cluster containing all data points. This raises the problem of how to chose the best clustering from this hierarchy. They solve this problem as follows: For each clustering result they evaluate the performance of the pruned ensemble on a validation set using majority voting as the combination method. The final pruned ensemble is the one that achieves the highest classification accuracy.

They experimented on a single data set, using heterogeneous classifiers derived by running different learning algorithms with different configurations. They compared their approach with *overproduce and choose* strategies and found that their approach exhibits better classification accuracy.

This approach is generally guided by the notion of diversity. Diversity guides both the clustering process and the subsequent pruning process. However, the authors use the classification accuracy with a specific combination method (majority voting) to select among the different clustering results. This reduces the generality of the method, as the selection is optimized towards majority voting. Of course this could be easily alleviated by using at that stage the method that will be later used for combining the ensemble.

In addition, the authors used a specific distance metric to guide the clustering process, while it would be interesting to evaluate the performance of other pairwise diversity metrics, like the ones proposed by Kuncheva [11]. Their limited (datasets) experimental results however does not guarantee the general utility of their method.

3.2.2 Lazarevic and Obradovic, 2001

Lazarevic and Obradovic [12] use the k-means algorithm to perform the clustering of classifiers. The k-means algorithm is applied to a table of data with as many rows as the classifiers and as many columns as the instances of the training set. The table contains the predictions of each classifier on each instance. Similar to HAC, the k-means algorithm suffers from the problem of selecting the number of clusters (k). The authors solve this problem, by considering iteratively a larger number of clusters until the diversity between them starts to decrease.

Subsequently, the authors prune the classifiers of each cluster using the following approach until the accuracy of the ensemble is decreased. They consider the classifiers in turn from the least accurate to the most accurate. A classifier is kept in the ensemble if its disagreement with the most accurate classifier is more than a predefined threshold and is sufficiently accurate. In addition to simple elimination of classifiers a method for distributing their voting weights is also implemented.

They experimented on four different data sets, using neural network ensembles produced with bagging and boosting. They compare the performance of their pruning method with that of unpruned ensembles and another ad-hoc method that they propose (see other methods) and find that their clustering-based approach offers the highest classification accuracy.

Their method suffers from the fact of parameter setting. How does one set the threshold for pruning models? In addition, the method is not compared to any other pruning methods and sufficient data sets, so its utility cannot be determined. It is very heuristic and ad-hoc.

3.2.3 Fu, Hu and Zhao, 2005

The work of [8] is largely based on the two previous methods. Similarly to [12] it uses the k-means algorithm for clustering the models of an ensemble. Similarly to [9] it prunes each cluster by selecting the single best performing model and uses the accuracy of the pruned ensemble to select the number of clusters.

The difference of this work with the other two clustering-based methods, is merely that the experiments are performed on regression data sets. However, both previous methods could be relatively easily extended to handle the pruning of regression models. The experiments of this work are performed on four data sets using an ensemble of neural networks produced with bagging and boosting, similar to [12].

3.3 Ranking-based

Ranking-based methods order the classifiers in the ensemble *once* according to some evaluation metric and select the classifiers in this order. They differ mainly in the criterion used for ordering the members of the ensemble.

A key concept in *Orientation Ordering* [15] is the *signature vector*. The signature vector of a classifier c is a $|D|$-dimensional vector with elements taking the value +1 if $c(x_i) = y_i$ and -1 if $c(x_i) \neq y_i$. The average signature vector of all classifiers in an ensemble is called the *ensemble signature vector* and is indicative of the ability of the Voting ensemble combination method to correctly classify each of the training examples. The *reference vector* is a vector perpendicular to the ensemble signature vector that corresponds to the projection of the first quadrant diagonal onto the hyper-plane defined by the ensemble signature vector.

In Orientation Ordering the classifiers are ordered by increasing values of the angle between their signature vector and the reference vector. Only the classifiers whose angle is less than $\pi/2$ are included in the final ensemble. Essentially this ordering gives preference to classifiers, which correctly classify those examples that are incorrectly classified by the full ensemble.

3.4 Other

This category includes two approaches that don't belong to any of the previous categories. The first one is based on statistical procedures for directly selecting a subset of classifiers, while the second is based on semi-definite programming.

Tsoumakas et al. [21, 20] 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. The obtained results are better than those of state-of-the-art ensemble methods.

Zhang et al. [24] formulate the ensemble pruning problem as a mathematical problem and apply semi-definite programming (SDP) techniques. In specific, the authors initially formulated the ensemble pruning problem as a quadratic integer programming problem that looks for a fixed-size subset of k classifiers with minimum misclassification and maximum divergence. They subsequently found that this quadratic integer programming problem is similar to the "max cut with size k" problem, which can be approximately solved using an algorithm based on SDP. Their algorithm requires the number of classifiers to retain as a parameter and runs in polynomial time.

4 Greedy Ensemble Selection

Greedy ensemble selection algorithms attempt to find the globally best subset of classifiers by taking local greedy decisions for changing the current subset. An example of the search space for an ensemble of four models is presented in Figure 1.

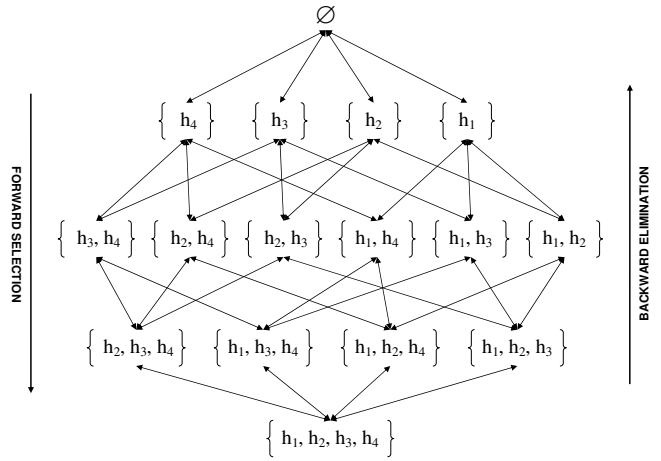

Figure 1. An example of the search space of greedy ensemble selection algorithms for an ensemble of four models.

In the following subsections we present and discuss on what we consider to be the main aspects of greedy ensemble selection algorithms: the direction of search, the measure and dataset used for evaluating the different branches of the search and the size of the final subensemble. The notation that will be used is the following.

- $D = \{(x_i, y_i), i = 1, 2, \ldots, N\}$ is an evaluation set of labelled training examples where each example consists of a feature vector x_i and a class label y_i.
- $H = \{h_t, t = 1, 2, \ldots, T\}$ is the set of classifiers or hypotheses of an ensemble, where each classifier h_t maps an instance x to a class label y, $h_t(x) = y$.
- $S \subseteq H$, is the current subensemble during the search in the space of subensembles.

4.1 Direction of Search

Based on the direction of search, there are two main categories of greedy ensemble selection algorithms: *forward selection* and *backward elimination*.

In forward selection, the current classifier subset S is initialized to the empty set. The algorithm continues by iteratively adding to S the classifier $h_t \in H \backslash S$ that optimizes an evaluation function $f_{FS}(S, h_t, D)$. This function evaluates the addition of classifier h_t in the current subset S based on the labelled data of D. For example, f_{FS} could return the accuracy of the ensemble $S \cup h_t$ on the data set D by combining the decisions of the classifiers with the method of voting. Algorithm 1 shows the pseudocode of the forward selection ensemble selection algorithm. In the past, this approach has been used in [7, 14, 4] and in the Reduce-Error Pruning with Backfitting (REPwB) method in [13].

In backward elimination, the current classifier subset S is initialized to the complete ensemble H and the algorithm continues by iteratively removing from S the classifier $h_t \in S$ that optimizes the evaluation function $f_{BE}(S, h_t, D)$. This function evaluates the removal of classifier h from the current subset S based on the labelled data of D. For example, f_{BE} could return a measure of diversity for the ensemble $S \setminus \{h_t\}$, calculated on the data of D. Algorithm 2 shows the pseudocode of the backward elimination ensemble selection algorithm. In the past, this approach has been used in the *AID thinning* and *concurrency thinning* algorithms [1].

Algorithm 1 The forward selection method in pseudocode

Require: Ensemble of classifiers H, evaluation function f_{FS}, evaluation set D
1: $S = \emptyset$
2: **while** $S \neq H$ **do**
3: $\quad h_t = \arg\max_{h \in H \setminus S} f_{FS}(S, h, D)$
4: $\quad S = S \cup \{h_t\}$
5: **end while**

Algorithm 2 The backward elimination method in pseudocode

Require: Ensemble of classifiers H, evaluation function f_{BE}, evaluation set D
1: $S = H$
2: **while** $S \neq \emptyset$ **do**
3: $\quad h_t = \arg\max_{h \in S} f_{BE}(S, h, D)$
4: $\quad S = S \setminus \{h_t\}$
5: **end while**

The time complexity of greedy ensemble selection algorithms for traversing the space of subensembles is $O(t^2 g(T, N))$. The term $g(T, N)$ concerns the complexity of the evaluation function, which is linear with respect to N and ranges from constant to quadratic with respect to T, as we shall see in the following subsections.

4.2 Evaluation Function

One of the main components of greedy ensemble selection algorithms is the function that evaluates the alternative branches during the search in the space of subensembles. Given a subensemble S and a model h_t the evaluation function estimates the utility of inserting (deleting) h_t into (from) S using an appropriate *evaluation measure*, which is calculated on an *evaluation dataset*. Both the measure and the dataset used for evaluation are very important, as their choice affects the quality of the evaluation function and as a result the quality of the selected ensemble.

4.2.1 Evaluation Dataset

One approach is to use the training dataset for evaluation, as in [14]. This approach offers the benefit that plenty of data will be available for evaluation and training, but is susceptible to the danger of overfitting.

Another approach is to withhold a part of the training set for evaluation, as in [4, 1] and in the REPwB method in [13]. This approach is less prone to overfitting, but reduces the amount of data that are available for training and evaluation compared to the previous approach. It sacrifices both the predictive performance of the ensemble's members and the quantity of the evaluation data for the sake of using unseen data in the evaluation. This method should probably be preferred over the previous one, when there is abundance of training data.

An alternative approach that has been used in [3], is based on k-fold cross-validation. For each fold an ensemble is created using the remaining folds as the training set. The same fold is used as the evaluation dataset for models and subensembles of this ensemble. Finally, the evaluations are averaged across all folds. This approach is less prone to overfitting as the evaluation of models is based on data that were not used for their training and at the same time, the complete training dataset is used for evaluation.

During testing the above approach works as follows: the k models that where trained using the same procedure (same algorithm, same subset, etc.) form a cross-validated model. When the cross-validated model makes a prediction for an instance, it averages the predictions of the individuals models. An alternative testing strategy that we suggest for the above approach is to train an additional single model from the complete training set and use this single model during testing.

4.2.2 Evaluation Measure

The evaluation measures can be grouped into two major categories: those that are based on *performance* and those on *diversity*.

The goal of performance-based measures is to find the model that maximizes the performance of the ensemble produced by adding (removing) a model to (from) the current ensemble. Their calculation depends on the method used for ensemble combination, which usually is voting. Accuracy was used as an evaluation measure in [13, 7], while [4] experimented with several metrics, including accuracy, root-mean-squared-error, mean cross-entropy, lift, precision/recall break-even point, precision/recall F-score, average precision and ROC area. Another measure is *benefit* which is based on a cost model and has been used in [7].

The calculation of performance-based metrics requires the decision of the ensemble on all examples of the pruning dataset. Therefore, the complexity of these measures is $O(|S|N)$. However, this complexity can be optimized to $O(N)$, if the predictions of the current ensemble are updated incrementally each time a classifier is added to/removed from it.

It is generally accepted that an ensemble should contain diverse models in order to achieve high predictive performance. However, there is no clear definition of diversity, neither a single measure to calculate it. In their interesting study, [11], could not reach into a solid conclusion on how to utilize diversity for the production of effective classifier ensembles. In a more recent theoretical and experimental study on diversity measures [19], the authors reached to the conclusion that diversity cannot be explicitly used for guiding the process of greedy ensemble selection. Yet, certain approaches have reported promising results [14, 1].

One issue that worths mentioning here is how to calculate the diversity during the search in the space of ensemble subsets. For simplicity we consider the case of forward selection only. Let S be the current ensemble and $h_t \in H \setminus S$ a candidate classifier to add to the ensemble.

One could compare the diversities of subensembles $S' = S \cup h_t$ for all candidate $h_t \in H \setminus S$ and select the ensemble with the highest diversity. Any pairwise and non-pairwise diversity measure can be used for this purpose. The time complexity of most non-pairwise diversity measures is $O(|S'|N)$, while that of pairwise diversity measures is $O(|S'|^2N)$. However, a straightforward optimization can be performed in the case of pairwise diversity measures. Instead of calculating the sum of the pairwise diversity for every pair of classifiers in each candidate ensemble S', one can simply calculate the sum of the pairwise diversities only for the pairs that include the candidate classifier h_t. The sum of the rest of the pairs is equal for all candidate ensembles. The same optimization can be achieved in backward elimination too. This reduces their time complexity to $O(|S|N)$.

Existing methods [14, 1, 19] use a different approach to calculate diversity during the search. They use pairwise measures to compare the candidate classifier h_t with the current ensemble S, which is viewed as a single classifier that combines the decisions of its members with voting. This way they calculate the diversity between the current ensemble as a whole and the candidate classifier. Such an approach has time complexity $O(|S|N)$, which can be optimized to $O(N)$, if the predictions of the current ensemble are updated incrementally each time a classifier is added to/removed from it. However, these calculations do not take into account the decisions of individual models.

In the past, the widely known diversity measures *disagreement*, *double fault*, *Kohavi-Wolpert variance*, *inter-rater agreement*, *generalized diversity* and *difficulty* were used for greedy ensemble selection in [19]. *Concurrency* [1], *margin distance minimization*, *Complementariness* [14] and *Focused Selection Diversity* are four diversity measures designed specifically for greedy ensemble selection. We next present these measures using a common notation. We can distinguish 4 events concerning the decision of the current ensemble and the candidate classifier:

$$\begin{aligned} e_1 &: y = h_t(\boldsymbol{x_i}) \wedge y \neq S(\boldsymbol{x_i}) \\ e_2 &: y \neq h_t(\boldsymbol{x_i}) \wedge y = S(\boldsymbol{x_i}) \\ e_3 &: y = h_t(\boldsymbol{x_i}) \wedge y = S(\boldsymbol{x_i}) \\ e_4 &: y \neq h_t(\boldsymbol{x_i}) \wedge y \neq S(\boldsymbol{x_i}) \end{aligned}$$

The *complementariness* of a model h_k with respect to a subensemble S and a set of examples $D = (x_i, y_i), i = 1, 2, \ldots, N$ is calculated as follows:

$$COM_D(h_k, S) = \sum_{i=1}^{N} I(e_1),$$

where $I(true) = 1$, $I(false) = 0$ and $S(x_i)$ is the classification of instance x_i from the subensemble S. This classification is derived from the application of an ensemble combination method to S, which usually is voting. The complementariness of a model with respect to a subensemble is actually the number of examples of D that are classified correctly by the model and incorrectly by the subensemble. A selection algorithm that uses the above measure, tries to add (remove) at each step the model that helps the subensemble classify correctly the examples it gets wrong.

The *concurrency* of a model h_k with respect to a subensemble S and a set of examples $D = (x_i, y_i), i = 1, 2, \ldots, N$ is calculated as follows:

$$CON_D(h_k, S) = \sum_{i=1}^{N} \Big(2 * I(e_1) + I(e_3) - 2 * I(e_4)\Big)$$

This measure is very similar to complementariness with the difference that it takes into account two extra cases.

The *focused ensemble selection* method [17] uses all the events and also takes into account the strength of the current ensemble's decision. Focused ensemble selection is calculated with the following

form:

$$FES(h_k, S) = \sum_{i=1}^{N} \Big(NT_i * I(e_1) - NF_i * I(e_2) + \\ + NF_i * I(e_3) - NT_i * I(e_4) \Big),$$

where NT_i denotes the proportion of models in the current ensemble S that classify example $(\boldsymbol{x_i}, y_i)$ correctly, and $NF_i = 1 - NT_i$ denotes the number of models in S that classify it incorrectly.

The *margin distance minimization* method [14] follows a different approach for calculating the diversity. For each classifier h_t an N-dimensional vector, c_t, is defined where each element $c_t(i)$ is equal to 1 if the t^{th} classifier classifies correctly instance i, and -1 otherwise. The vector, C_S of the subensemble S is the average of the individual vectors c_t, $C_S = \frac{1}{|S|} \sum_{t=1}^{|S|} c_t$. When S classifies correctly all the instances the corresponding vector is in the first quadrant of the N-dimensional hyperplane. The objective is to reduce the distance, $d(o, C)$, where d is the Euclidean distance and o a predefined vector placed in the first quadrant. The margin, $MAR_D(h_k, S)$, of a classifier k with respect to a subensemble S and a set of examples $D = (x_i, y_i), i = 1, 2, \ldots, N$ is calculated as follows:

$$MAR_D(h_k, S) = d\left(o, \frac{1}{|S| + 1} \Big(c_k + C_S \Big) \right)$$

4.3 Size of Final Ensemble

Another issue that concerns greedy ensemble selection algorithms, is when to stop the search process, 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 subensemble 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 [13, 7, 14, 1].

5 Conclusions

This works was a first attempt towards a taxonomy of ensemble selection methods. We believe that such a taxonomy is necessary for researchers working on new methods. It will help them identify the main categories of methods and their key points, and avoid duplication of work. Due to the large amount of existing methods and the different parameters of an ensemble selection framework (heterogeneous/homogeneous ensemble, algorithms used, size of ensemble, etc), it is possible to devise a new method, which may only differ in small, perhaps unimportant, details from existing methods. A generalized view of the methods, as offered from a taxonomy, will help avoid work towards such small differences, and perhaps may lead to more novel methods.

Of course, we do not argue that the proposed taxonomy is perfect. On the contrary, it is just a first and limited step in abstracting and categorizing the different methods. Much more elaborate study has to be made, to properly account for the different aspects of existing methods. No doubt, some high quality methods may have been left outside this study. We hope that through a discussion and the criticism of this work within the ensemble methods community, and especially people working on ensemble selection, a much improved version of it will arise.

REFERENCES

[1] R. E. Banfield, L. O. Hall, K. W. Bowyer, and W. P. Kegelmeyer, 'Ensemble diversity measures and their application to thinning.', *Information Fusion*, **6**(1), 49–62, (2005).

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

[3] R. Caruana, A. Munson, and A. Niculescu-Mizil, 'Getting the most out of ensemble selection', in *Sixth International Conference in Data Mining (ICDM '06)*, (2006).

[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*, p. 18, (2004).

[5] T. G. Dietterich, 'Machine-learning research: Four current directions', *AI Magazine*, **18**(4), 97–136, (1997).

[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] W. Fan, F. Chu, H. Wang, and P. S. Yu, 'Pruning and dynamic scheduling of cost-sensitive ensembles', in *Eighteenth national conference on Artificial intelligence*, pp. 146–151. American Association for Artificial Intelligence, (2002).

[8] Qiang Fu, Shang-Xu Hu, and Sheng-Ying Zhao, 'Clusterin-based selective neural network ensemble', *Journal of Zhejiang University SCIENCE*, **6A**(5), 387–392, (2005).

[9] Giorgio Giacinto, Fabio Roli, and Giorgio Fumera, 'Design of effective multiple classifier systems by clustering of classifiers', in *15th International Conference on Pattern Recognition, ICPR 2000*, pp. 160–163, (3–8 September 2000).

[10] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, 'Adaptive mixtures of local experts', *Neural Computation*, **3**, 79–87, (1991).

[11] L.I. Kuncheva and C.J. Whitaker, 'Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy', *Machine Learning*, **51**, 181–207, (2003).

[12] Aleksandar Lazarevic and Zoran Obradovic, 'Effective pruning of neural network classifiers', in *2001 IEEE/INNS International Conference on Neural Networks, IJCNN 2001*, pp. 796–801, (15–19 July 2001).

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

[14] 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).

[15] 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).

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

[17] I. Partalas, G. Tsoumakas, and I. Vlahavas, 'Focused ensemble selection: A diversity-based method for greedy ensemble selection', in *18th European Conference on Artificial Intelligence*, (2008).

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

[19] E. K. Tang, P. N. Suganthan, and X. Yao, 'An analysis of diversity measures', *Machine Learning*, **65**(1), 247–271, (2006).

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

[21] G. Tsoumakas, I. Katakis, and I. Vlahavas, 'Effective Voting of Heterogeneous Classifiers', in *Proceedings of the 15th European Conference on Machine Learning, ECML2004*, pp. 465–476, (2004).

[22] C.J. Watkins and P. Dayan, 'Q-learning', *Machine Learning*, **8**, 279–292, (1992).

[23] D. Wolpert, 'Stacked generalization', *Neural Networks*, **5**, 241–259, (1992).

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

[25] 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).