Taxonomy for Characterizing Ensemble Methods in Classification Tasks: a review and annotated bibliography

Lior Rokach
Department of Information System Engineering
Ben-Gurion University of the Negev
liorrk@bgu.ac.il

Abstract

Ensemble methodology, which builds a classification model by integrating multiple classifiers, can be used for improving prediction performance. Researchers from various disciplines such as statistics, pattern recognition, and machine learning have seriously explored the use of ensemble methodology. This paper presents an updated survey of ensemble methods in classification tasks, while introducing a new taxonomy for characterizing them. The new taxonomy, presented from the algorithm designer's point of view, is based on five dimensions: inducer, combiner, diversity, size, and members dependency. We also propose several selection criteria, presented from the practitioner's point of view, for choosing the most suitable ensemble method.

Key words:
Ensemble-methods; Classification; Boosting; Bagging; Partitioning; Decision trees; Neural networks

1. Introduction

Supervised learning methods are methods that attempt to discover relationships between the input attributes (independent variables) and the target attribute (dependent variable). The relationship discovered is represented in a structure referred to as a model. Usually models can be used for predicting the value of the target attribute knowing the values of the input attributes. It is useful to distinguish between two main supervised models: classification models (classifiers) and regression models.

Regression models map the input space into a real-valued domain, whereas classifiers map the input space into predefined classes. For instance, classifiers can be used to classify mortgage consumers into good (fully payback the mortgage on time) and bad (delayed payback).

In a typical supervised learning problem, a training set of labeled examples is given and the goal is to form a description that can be used to predict previously

unseen examples.

The main idea of an ensemble methodology is to combine a set of models, each of which solves the same original task, in order to obtain a better composite global model, with more accurate and reliable estimates or decisions than can be obtained from using a single model.

In the literature, the term "ensemble methods" is usually reserved for collections of models that are minor variants of the same basic model. Nevertheless, in this survey we also cover hybridization of models that are not from the same family. The later is also referred in the literature as "multiple classifier systems" [69, 67].

In fact, ensemble methodology imitates our second nature to seek several opinions before making any crucial decision. We weigh the individual opinions, and combine them to reach a final decision [121].

The ensemble methodology has been used to improve the predictive performance of single models, in many fields such as: finance [92], bioinformatics [162], medicine [103], cheminformatics [108], manufacturing [130, 131, 102], geography [26], information security [106, 113] Information Retrieval [51, 52, 142, 9, 142], Image Retrieval [95, 163] and recommender systems [148].

The idea of building a predictive model by integrating multiple models has been under investigation for a long time. The history of ensemble methods starts as early as 1977 with Tukeys Twicing [170], an ensemble of two linear regression models. Tukey suggested to fit the first linear regression model to the original data and the second linear model to the residuals. Two years later, Dasarathy and Sheela [39] suggest to partition the input space using two or more classifiers. However the main progress in the field has been made during the Nineties. Hansen and Salamon [64] suggested an ensemble of similarly configured neural networks to improve the predictive performance of a single ANN. At the same time Schapire [146] laid the foundations for the award winning AdaBoost [55] algorithm by showing that a strong classifier in probably approximately correct (PAC) sense can be generated by combining "weak" classifiers (that is, simple classifiers whose classification power is only slightly better than random classification). In this paper we restrict attention to classification tasks although the ensemble methodology can be used in other tasks such as regression [61, 174, 82] or density estimation [128].

In the past few years, experimental studies conducted by the machine-learning community show that combining the outputs of multiple classifiers reduces the generalization error [48, 124, 14, 116]. Ensemble methods are very effective, mainly due to the phenomenon that various types of classifiers have different "inductive biases" [112]. Indeed, ensemble methods can effectively make use of such diversity to reduce the variance-error [171, 4] without increasing the bias-error. In certain situations, an ensemble can also reduce bias-error, as shown by the theory of large margin classifiers [13].

2. Existing Surveys on Ensemble of Classifiers

Given the potential usefulness of ensemble methods, it is not surprising that a vast number of methods are now available to researchers and practitioners. The variety of ensemble techniques have arisen several taxonomies in the literature which aim to categorize ensemble methods from the algorithm designer point of view.

Sharkey [155] proposed a taxonomy for ensemble of neural networks. This taxonomy suggests three dimensions:

1. The first dimension indicates if the ensemble's members are competitive or cooperative. In the competitive mode, a single member is selected to provide the classification. In cooperative mode the classifications of all members are combined.
2. The second dimension indicates if the ensemble is created top-down or bottom-up. In top-down mode the combination mechanism is based on something other than the classifiers outputs. Bottom-up techniques take the outputs of the members into account in their combination. Bottom up methods are subdivided into fixed methods (such as voting), and dynamic methods (such as stacking). In fixed methods, although the outputs are implicated in the computation, the method of combining remains fixed. In dynamic methods, the relative contribution of component classifiers varies as a function of their output.
3. The third dimension indicates if we combine either ensemble, modular, or hybrid components; Sharkey [152] and Lam [90] distinguish between modular systems and pure ensemble systems. The main idea of pure ensemble systems is to combine a set of classifiers, each of which solves the same original task. The purpose of pure ensemble systems is to obtain a more accurate and reliable performance than using a single classifier. On the other hand, the purpose of modular systems is to break down a complex problem into several manageable problems such that each inducer is used to either solve a different task or it is applied to a different training set schema.

Ho [66] as well as Valentini and Masulli [175] dichotomized the ensemble techniques into two main categories:

1. decision optimization methods (such as mixture of experts) — Use a fixed set of carefully designed and highly specialized classifiers. The goal of decision optimization methods is to find an optimal combination of their classifications.
2. coverage optimization (such as boosting) — use a fixed decision combination function. Coverage optimization methods generate a set of mutually complementary, generic classifiers that are combined to improve predictive performance.

Brown et al. [25] divides up the ensemble methods according to whether they choose to implicitly obtain diversity by randomization methods or whether

they explicitly gain diversity via some metric. They then grouped techniques according to three factors: how they initialize the inducers in the hypothesis space, what the space of accessible hypotheses is, and how that space is traversed by the inducer.

Kuncheva [86] proposes to group ensemble methods according to the ways these ensembles are built. Specifically, there are four layers:

1. Combination level - defines different ways of combining the classifier decisions.
2. Classifier level - indicates which base classifiers are used to constitute the ensemble.
3. Feature level - in this level different feature subsets can be used for the classifiers.
4. Data level - indicates which dataset is used to train each base classifier.

Duin [50] as well as Kamel and Wanas [80] propose to differentiate between trainable and nontrainable ensembles. Specifically nontrainable ensembles do not need training after the base classifiers have been induced. Trainable ensembles need additional training to create the ensemble (either during the base classifier training or after all base classifiers are trained).

Although several surveys on ensemble for classification tasks are available in the literature [46, 121, 25] and there are several papers which suggest a taxonomy for ensemble methods [25, 66, 155], in this paper we introduce four main contributions:

1. We suggest a new unified taxonomy to categorize all significant ensemble methods developed in the field. As indicated in [86]: "We still do not have an agreed upon structure or a taxonomy of the whole field, although a silhouette of a structure is slowly crystallizing among the numerous attempts." On the one hand because existing taxonomies usually concentrate on some aspects (for example [25] concentrates on diversity), the new proposed taxonomy tries to organize existing taxonomies into a coherent and unified taxonomy. On the other hand the new taxonomy introduce new elements that have not proposed before. The goal of the new taxonomy is to better distinguish between existing ensemble methods and to help algorithm designers to identify new opportunities (i.e. combinations that have not been deeply explored).
2. Due to the fact that ensemble learning is an active research field, this paper proposes an updated survey which refers to new researches from the last three years that have not been previously covered by existing surveys.
3. The proposed paper covers efficient and mature ensemble methods that do not belong to the mainstream, and therefore are usually not mentioned in existing surveys. For example DECORATE [105], Arbiter Trees [30] and attribute bagging [23].
4. We also propose several selection criteria, presented from the practitioner's point of view, for choosing the most suitable ensemble method.

3. Overview of the proposed taxonomy

A typical ensemble method for classification tasks contains the following building blocks:

1. Training set - A labeled dataset used for training the ensemble. Most frequently the training set is a collection of instances (also known as samples or observations). Each instance is described by an attribute-value vectors. The input space is spanned by the attributes used to describe the instances. In semi-supervised methods of ensemble generation, such as ASSEMBLE [16], unlabeled instances can be also used for the creation of the ensemble.
2. Inducer – The inducer is an induction algorithm that obtains a training set and forms a classifier that represents the generalized relationship between the input attributes and the target attribute.
3. Ensemble generator – This component is responsible for generating the diverse classifiers.
4. Combiner - The combiner is responsible for combining the classifications of the various classifiers.

The nature of each building block and especially the relation among them can be used to categorize ensemble methods. Our taxonomy consists of the following dimensions:

1. Combiner usage — This property specifies the relation between the ensemble generator and the combiner.
2. Classifiers dependency — During the classifier training how does each classifier affect the other classifiers? Classifiers may be dependent or independent.
3. Diversity generator — In order to make the ensemble more effective, there should be some sort of diversity between the classifiers [87]. Brown et al. [25] indicate that for classification tasks the concept of "diversity" is still an ill-defined concept. Nevertheless it is believed to be closely related to the statistical concept of correlation. Diversity is obtained when the misclassification events of the base classifiers are not correlated. Several means can be used to reach this goal: different presentations of the input data, variations in learner design, or by adding a penalty to the outputs to encourage diversity.
4. Ensemble size — The number of classifiers in the ensemble and how the undesirable classifiers are removed from the ensemble.
5. Cross-Inducer — A Cross-inducer ensemble techniques could run on all common inducers, or simply more than one. Some ensembles have been specifically designed for a certain inducer and can not be used for other inducers.

The issues of classifiers' dependency and diversity are closely linked. More specifically, it can be argued that any effective method for generating diversity results in dependent classifiers (otherwise obtaining diversity is just luck).

Nevertheless, it is very simple to independently build ensemble of diverse classifiers. For example, we can train in parallel a number of different classifiers by randomly drawing their training instances from the original set. Because none of these classifiers in anyway affects the training of others, they remain "independent". Moreover , as we will explain later one can independently create the classifiers and then, as a post-processing step, select the most diverse classifiers. Instead of using the notion of classifiers' dependency, Brown et al. [25] make a distinction between explicit and implicit diversity methods. In implicit techniques no measurement is taken to ensure diversity will emerge (thus, the classifiers can be independently trained). Explicit methods, on the other hand, explicitly try to optimize some metric of diversity during building the ensemble (thus, the classifiers are usually build in some dependent manner as they need together maximize diversity).

Naturally there might be other dimensions which can be used to differentiate an ensemble scheme and we discuss it in the conclusion section.

The rest of this paper is organized as follows: In sections 3 to 8 we discuss and describe each one of the above mentioned dimensions in details. Section 9 illustrates how the new taxonomy can be used to categorize popular ensemble methods. Section 10 suggests criteria for selecting an ensemble method from the practitioner point of view. Finally, Section 11 concludes the work and presents suggestions for further research in the field.

4. Combiner Usage

This property specifies the relation between the ensemble generator and the combiner. Some ensemble generators are combiner-dependent. That is to say, they have been developed specifically for a certain combination method. Other ensemble generators are combiner-independent; the combination method is provided as input to the framework. Potentially there could be ensemble generators that, given a set of combiners, would be capable of choosing the best combiner in the current case.

There are two main methods for combining classifiers: weighting methods and meta-learning. The weighting methods are best suited for problems where the individual classifiers perform the same task and have comparable success or when we would like to avoid problems associated with added learning (such as overfitting or long training time). Meta-learning methods are best suited for cases in which certain classifiers consistently correctly classify, or consistently misclassify, certain instances.

4.1. Weighting Methods

When combining classifiers with weights, a classifier's classification has a strength proportional to its assigned weight. The assigned weight can be fixed or dynamically determined for the specific instance to be classified. The following weighting methods are frequently used in the literature:

Majority Voting Select the class that obtains the highest number of votes among the ensemble members [4].

Mathematically, majority voting can be written as:

$$class(x) = \underset{c_i \in dom(y)}{\arg\max} \left(\sum_k g\left(y_k(x), c_i\right) \right) \tag{1}$$

where $y_k(x)$ is the classification of the k'th classifier and $g(y, c)$ is an indicator function defined as:

$$g\left(y, c\right) = \begin{cases} 1 & y = c \\ 0 & y \neq c \end{cases} \tag{2}$$

Note that in case of a probabilistic classifier, the crisp classification $y_k(x)$ is usually obtained as follows:

$$y_k(x) = \underset{c_i \in dom(y)}{\arg\max} \hat{P}_{M_k}(y = c_i \,|x) \tag{3}$$

where M_k denotes classifier k and $\hat{P}_{M_k}(y = c \,|x\,)$ denotes the probability of y obtaining the value c given an instance x.

Performance Weighting Weight the classifiers proportionally to their accuracy performance on a validation set [117].

Mathematically, performance weighting can be written as:

$$\alpha_i = \frac{(1 - E_i)}{\sum_{j=1}^{T} (1 - E_j)} \tag{4}$$

where E_i is a normalization factor which is based on the performance evaluation of classifier i on a validation set.

Distribution Summation Sum up the conditional probability vector obtained from each classifier, and select the most probable class [35]. Mathematically, it can be written as:

$$Class(x) = \underset{c_i \in dom(y)}{\operatorname{argmax}} \sum_k \hat{P}_{M_k}(y = c_i \,|x\,) \tag{5}$$

Dempster–Shafer Use Dempster–Shafer theory of evidence for combining classifiers [156]. This method uses the notion of basic probability assignment defined for a certain class c_i given the instance x:

$$bpa(c_i, x) = 1 - \prod_k \left(1 - \hat{P}_{M_k}(y = c_i \,|x\,)\right) \tag{6}$$

Consequently, the selected class is the one that maximizes the value of the belief function:

$$Bel(c_i, x) = \frac{1}{A} \cdot \frac{bpa(c_i, x)}{1 - bpa(c_i, x)} \tag{7}$$

where A is a normalization factor defined as:

$$A = \sum_{\forall c_i \in dom(y)} \frac{bpa(c_i, x)}{1 - bpa(c_i, x)} + 1 \tag{8}$$

Vogging (Variance Optimized Bagging) Optimize a linear combination of base-classifiers so as to aggressively reduce variance while attempting to preserve a prescribed accuracy [42].

Naïve Bayes Using Naïve Bayes for combining various classifiers assuming the classifiers predictions are conditionally independent given the class [138]:

$$Class(x) = \underset{\substack{c_j \in dom(y) \\ \hat{P}(y=c_j) > 0}}{\operatorname{argmax}} \hat{P}(y = c_j) \cdot \prod_{k=1} \frac{\hat{P}_{M_k}(y = c_j \,|x\,)}{\hat{P}(y = c_j)} \tag{9}$$

Logarithmic Opinion Pool Extending the Naïve Bayes combination by providing a different weight for each member [63]:

$$Class(x) = \underset{c_j \in dom(y)}{\operatorname{argmax}} e^{\sum_k \alpha_k \cdot \log(\hat{P}_{M_k}(y=c_j\,|x\,))} \tag{10}$$

where α_k denotes the weight of the k-th classifier, such that:

$$\alpha_k \geq 0; \sum \alpha_k = 1 \tag{11}$$

DEA Using the data envelop analysis (DEA) methodology in order to assign weights to different classifiers [158].

Gating Network - Use a gating network to adjust the weights of the classifiers based on the input to be classified [77].

Some of the weighting methods are trainable. Lin et al. [96] propose to use genetic algorithms in attempt to find the optimal weights. They describe two different combinatory schemes to improve the performance of handwritten Chinese character recognition: the accuracy rate of the first candidate class and the accuracy rate of top ten candidate classes. Their extensive study show that this new approach can significantly improve the accuracy performance.

Reinforcement learning (RL) has been used to adaptively combine the base classifiers [47]. The ensemble consists of a controlling agent that selects which base classifiers are used to classify a particular instance. The controlling agent learn to make decisions so that classification error is minimized. The agent is trained through a Q-learning inspired technique. The usage of reinforcement learning improves results when there are many base classifiers.

4.2. Meta-combination Methods

Meta-learning means learning from the classifiers produced by the inducers and from the classifications of these classifiers on training data. The following meta-combination methods are frequently used in the literature:

Stacking The idea is to create a meta-dataset containing a instance for each instance in the original dataset [183, 43]. However, instead of using the original input attributes, it uses the predicted classifications by the classifiers as the input attributes. The target attribute remains as in the original training set. A test instance is first classified by each of the base classifiers. These classifications are used to create a meta-level training set from which a meta-classifier is produced. This classifier combines the different predictions into a final one. Stacking is usually employed to combine models built by different inducers. There are several extensions to the stacking approach, including StackingC [150], Troika [107] and SCANN (Stacking, Correspondence Analysis and Nearest Neighbor) [110].

Arbiter Trees In this method, the training set is randomly partitioned into k disjoint subsets [30]. The arbiter is induced from a pair of classifiers and recursively a new arbiter is induced from the output of two arbiters. Consequently for k classifiers, there are $\log_2(k)$ levels in the arbiter tree which is built in a bottom-up fashion.

The creation of the arbiter is performed as follows. For each pair of classifiers, the union of their training dataset is classified by the two classifiers. A selection rule compares the classifications of the two classifiers and selects instances from the union set to form the training set for the arbiter. The arbiter is induced from this set with the same learning algorithm used in the base level. The purpose of the arbiter is to provide an alternate classification when the base classifiers present diverse classifications. This arbiter, together with an arbitration rule, decides on a final classification outcome, based upon the base predictions. The process of forming the union of data subsets; classifying it using a pair of arbiter trees; comparing the classifications; forming a training set; training the arbiter; and picking one of the predictions, is recursively performed until the root arbiter is formed.

Combiner Trees This is an extension to the arbiter trees, in which a combiner, instead of an arbiter, is placed in each non-leaf node of a combiner tree [31]. In the combiner strategy, the classifications of the learned base classifiers form the basis of the meta-learner's training set. A composition rule determines the content of training examples from which a combiner (meta-classifier) will be generated. In classifying an instance, the base classifiers first generate their classifications and based on the composition rule, a new instance is generated. The aim of this strategy is to combine the classifications from the base classifiers by learning the relationship between these classifications and the correct classification. Hothorn and

Lausen [70] propose to use classification tree to fuse the classifications of arbitrary classifiers. The classification tree is trained on the original input attributes and additional transformations of the original attributes. The transformations are non-linear functions which are trained using either the out-of-bag samples or additional bootstrap samples as learning samples. Similarly Buttrey and Karo [28] incorporate nearest neighbors in the leaves of a tree.

Grading This technique uses "graded" classifications as meta-level classes [149]. The term "graded" is used in the sense of classifications that have been marked as correct or incorrect. The method transforms the classification made by the k different classifiers into k training sets by using the instances k times and attaching them to a new binary class in each occurrence. This class indicates whether the k–th classifier yielded a correct or incorrect classification, compared to the real class of the instance.

For each base classifier, one meta-classifier is learned whose task is to classify when the base classifier will misclassify. At classification time, each base classifier classifies the unlabeled instance. The final classification is derived from the classifications of those base classifiers that are classified to be correct by the meta-classification schemes. In case several base classifiers with different classification results are classified as correct, voting, or a combination considering the confidence estimates of the base classifiers, is performed. Grading may be considered as a generalization of cross-validation selection [145], which divides the training data into k subsets, builds $k-1$ classifiers by dropping one subset at a time and then uses it to find a misclassification rate. Finally, the procedure simply chooses the classifier corresponding to the subset with the smallest misclassification. Grading tries to make this decision separately for each and every instance by using only those classifiers that are predicted to classify that instance correctly. The main difference between grading and combiners (or stacking) is that the former does not change the instance attributes by replacing them with class predictions or class probabilities (or adding them to it). Instead it modifies the class values. Furthermore, in grading several sets of meta-data are created, one for each base classifier. Several meta-level classifiers are learned from those sets.

Adaptive fusion and co-operative training This hierarchical architecture uses a set of classifiers, called detectors, that are applied to make the aggregation scheme a more adaptive process. The detectors utilize both the input space and the prediction of the classifiers to present a weighting factor for each classifier. The weights and the output of the classifiers were then combined and used as inputs to the aggregation module. The latter learned how to combine different classifiers to produce the final decision. Furthermore, a co-operative training algorithm helps to determine whether further training is required. Applying the co-operative training algorithm, reduce the number of training epochs [177].

Mathematical Programming Adem and Gochet [2] formulate the combination task as a mixed integer linear programming problem. This formulation guarantees that the combined classification will be at least as good as the best base classifier. Moreover using this approach the user is capable to include extra problem-specific constraints into the combination task.

Most of the combination methods do not consider dependence relationship among base classifiers. Therefore, when the classifiers are highly dependent, the performance of these combining methods tend to be degraded and biased [81]. Behavior-Knowledge Space (BKS) method search for optimal combination without an independence assumption [73]. The BKS method determines a representative class, as a combined decision, which has maximum frequencies for given observed decisions. However the Achilles' heel of BKS is that it requires storage that is exponential to the ensemble size. In order to solve this drawback Kang and Lee [81] propose minimizing the upper bound of Bayes error rate. For this purpose they go beyond the first-order dependency assumption but avoiding the sparsity problem encountered by the BKS method.

Liu [98] proposes to transform the classifiers outputs to confidence measures. Each confidence transformation method is the combination of a scaling function (global normalization, Gaussian density modeling, and logistic regression) and a confidence type (linear, sigmoid, and evidence). The combination rules include fixed rules (sum-rule, product-rule, median-rule, etc.) and trained rules (linear discriminants and weighted combination with various parameter estimation techniques). The experimental study indicates that confidence transformation improves the accuracy performance of either fixed rules or trained rules. Trained rules mostly outperform fixed rules, especially when the classifier set contains weak classifiers.

5. Classifier Dependency

This property indicates whether the various classifiers are dependent or independent. In a dependent framework the outcome of a certain classifier affects the creation of the next classifier. Alternatively each classifier is built independently and their results are combined in some fashion. Some researchers refer to this property as "the relationship between modules" and distinguish between three different types: successive, cooperative and supervisory [152]. Roughly speaking, "successive" refers to "dependent" while "cooperative" refers to "independent". The last type applies to those cases in which one model controls the other model.

5.1. Dependent Methods

In dependent approaches for learning ensembles, there is an interaction between the learning runs. Thus it is possible to take advantage of knowledge generated in previous iterations to guide the learning in the next iterations. We distinguish between two main approaches for dependent learning, as described in the following sections [123].

5.1.1. Model-guided Instance Selection

In this dependent approach, the classifiers that were constructed in previous iterations are used for manipulating the training set for the following iteration. One can embed this process within the basic learning algorithm. These methods usually ignore all data instances on which their initial classifier is correct and only learn from misclassified instances.

The most well known model-guided instance selection is boosting. Boosting (also known as arcing — Adaptive Resampling and Combining) is a general method for improving the performance of a weak learner (such as classification rules or decision trees). The method works by repeatedly running a weak learner (such as classification rules or decision trees), on various distributed training data. The classifiers produced by the weak learners are then combined into a single composite strong classifier in order to achieve a higher accuracy than the weak learner's classifiers would have had.

The AdaBoost algorithm was first introduced in [55]. The main idea of this algorithm is to assign a weight in each example in the training set. In the beginning, all weights are equal, but in every round, the weights of all misclassified instances are increased while the weights of correctly classified instances are decreased. As a consequence, the weak learner is forced to focus on the difficult instances of the training set. This procedure provides a series of classifiers that complement one another.

Breiman [19] explores a simpler arcing algorithm called Arc-x4 which was aim to demonstrate that AdaBoost works not because of the specific form of the weighing function, but because of the adaptive resampling. In Arc-x4 the classifiers are combined by simple voting.

The basic AdaBoost algorithm deals with binary classification. Freund and Schapire describe two versions of the AdaBoost algorithm (AdaBoost.M1, AdaBoost.M2), which are equivalent for binary classification and differ in their handling of multi-classes classification problems.

Friedman et al. [57] present a generalized version of AdaBoost, which they call Real AdaBoost. The revised algorithm combines the class probability estimate of the classifiers by fitting an additive logistic regression model in a forward stepwise manner. The revision reduces computation cost and may lead to better performance especially in decision trees. Moreover it can provide interpretable descriptions of the aggregate decision rule.

Friedman [58] developed gradient boosting which builds ensemble by sequentially fitting base learner parameters to current "pseudo"-residuals by least squares at each iteration. The pseudo-residuals are the gradient of the loss functional being minimized, with respect to the model values at each training data point evaluated at the current step. To improve accuracy performance, increase robustness and reduce computational cost, at each iteration a subsample of the training set is randomly selected (without replacement) and used to fit the base classifier.

Phama and Smeuldersb [120] present a strategy to improve the AdaBoost algorithm with a quadratic combination of base classifiers. The idea is to con-

struct an intermediate learner operating on the combined linear and quadratic terms. First a classifier is trained by randomizing the labels of training examples. Then, the input learner is called repeatedly with a systematic update of the labels of the training examples in each round. This method is in contrast to the AdaBoost algorithm that uses reweighting of training examples. Together they form a powerful combination that makes intensive use the given base learner by both reweighting and relabeling the original training set.

Tsao and Chang [167] refer to boosting as a stochastic approximation procedure Based on this viewpoint they develop the SABoost (stochastic approximation) algorithm which is similar to AdaBoost except the way members' weights is calculated.

All boosting algorithms presented here assume that the weak inducers which are provided can cope with weighted instances. If this is not the case, an unweighted dataset is generated from the weighted data by a resampling technique. Namely, instances are chosen with a probability according to their weights (until the dataset becomes as large as the original training set).

Boosting seems to improve performance for two main reasons:

1. It generates a final classifier whose error on the training set is small by combining many hypotheses whose error may be large.
2. It produces a combined classifier whose variance is significantly lower than those produced by the weak learner.

On the other hand, boosting sometimes leads to a deterioration in generalization performance. According to Quinlan [124], the main reason for boosting's failure is overfitting. The objective of boosting is to construct a composite classifier that performs well on the data, but a large number of iterations may create a very complex composite classifier, that is significantly less accurate than a single classifier. A possible way to avoid overfitting is by keeping the number of iterations as small as possible. It has been shown that boosting approximates a large margin classifier such as the SVM [143].

AdaBoost rarely suffers from overfitting problems. Nevertheless in highly noisy datasets overfitting does occur. Sun et al. [161] pursue a strategy which penalizes the data distribution skewness in the learning process to prevent several hardest examples from spoiling decision boundaries. They use two smooth convex penalty functions, based on Kullback–Leibler divergence (KL) and l2 norm, to derive two new algorithms: AdaBoostKL and AdaBoostNorm2. These two AdaBoost variations achieve better performance on noisy datasets.

An online boosting algorithm called ivoting trains the base models using consecutive subsets of training examples of some fixed size [20]. For the first base classifier, the training instances are randomly selected from the training set. To generate a training set for the kth base classifier, ivoting selects a training set in which half the instances have been correctly classified by the ensemble consisting of the previous base classifiers and half have been misclassified. ivoting is an improvement on boosting that is less vulnerable to noise and overfitting. Further, since it does not require weighting the base classifiers, ivoting can be used in a parallel fashion, as demonstrated in [32].

Another important drawback of boosting is that it is difficult to understand. The resulting ensemble is considered to be less comprehensible since the user is required to capture several classifiers instead of a single classifier. Despite the above drawbacks, Breiman [18] refers to the boosting idea as the most significant development in classifier design of the Nineties.

Zhang and Zhang [188] have recently proposed a new boosting-by-resampling version of Adaboost. In the local Boosting algorithm, a local error is calculated for each training instance which is then used to update the probability that this instance is chosen for the training set of the next iteration. Classifying a new instance is based on its similarity with each training instance.

Merler et al. [109] developed the P-AdaBoost algorithm which is a distributed version of AdaBoost. Instead of updating the "weights" associated with instance in a sequential manner, P-AdaBoost works in two phases. In the first phase, the AdaBoost algorithm runs in its sequential, standard fashion for a limited number of steps. In the second phase the classifiers are trained in parallel using weights that are estimated from the first phase. P-AdaBoost yields approximations to the standard AdaBoost models that can be easily and efficiently distributed over a network of computing nodes.

5.1.2. Incremental Batch Learning

In this method the classification produced in one iteration is given as "prior knowledge" to the learning algorithm in the following iteration. The learning algorithm uses the current training set together with the classification of the former classifier for building the next classifier. The classifier constructed at the last iteration is chosen as the final classifier.

5.2. Independent Methods

In this methodology the original dataset is partitioned into several subsets from which multiple classifiers are induced. The subsets created from the original training set may be disjointed (mutually exclusive) or overlapping. A combination procedure is then applied in order to produce a single classification for a given instance. Since the method for combining the results of induced classifiers is usually independent of the induction algorithms, it can be used with different inducers at each subset. Moreover this methodology can be easily parallelized. These independent methods aim either at improving the predictive power of classifiers or decreasing the total execution time. The following sections describe several algorithms that implement this methodology.

5.2.1. Bagging

The most well-known independent method is bagging (bootstrap aggregating) [18]. The method aims to increase accuracy by creating an improved composite classifier, by amalgamating the various outputs of learned classifiers into a single prediction. Each classifier is trained on a sample of instances taken with a replacement from the training set. Each sample size is equal to the size of the original training set. Note that since sampling with replacement is used,

some of the original instances may appear more than once in the each training set and some may not be included at all.

So while the training sets may be different from each other, they are certainly not independent from a statistical point of view. To classify a new instance, each classifier returns the class prediction for the unknown instance. The composite bagged classifier, returns the class that has been predicted most often (voting method). The result is that bagging produces a combined model that often performs better than the single model built from the original single data. Breiman [18] notes that this is true especially for unstable inducers because bagging can eliminate their instability. In this context, an inducer is considered unstable if perturbing the learning set can cause significant changes in the constructed classifier.

Bagging, like boosting, is a technique for improving the accuracy of a classifier by producing different classifiers and combining multiple models. They both use a kind of voting for classification in order to combine the outputs of the different classifiers of the same type. In boosting, unlike bagging, each classifier is influenced by the performance of those built before with the new classifier trying to pay more attention to errors that were made in the previous ones and to their performances. In bagging, each instance is chosen with equal probability, while in boosting, instances are chosen with a probability proportional to their weight. Furthermore, according to Quinlan [124], as mentioned above, bagging requires that the learning system should not be stable, where boosting does not preclude the use of unstable learning systems, provided that their error rate can be kept below 0.5.

5.2.2. Wagging

Wagging is a variant of bagging [14] in which each classifier is trained on the entire training set, but each instance is stochastically assigned a weight.

In fact bagging can be considered to be wagging with allocation of weights from the Poisson distribution (each instance is represented in the sample a discrete number of times). Alternatively it is possible to allocate the weights from the exponential distribution, because the exponential distribution is the continuous valued counterpart to the Poisson distribution [179].

5.2.3. Random Forest and Random Subspace Projection

A Random Forest ensemble [21] uses a large number of individual, unpruned decision trees. The individual trees are constructed using a simple algorithm. The decision tree is not pruned and at each node, rather than choosing the best split among all attributes, the inducer randomly samples N (where N is an input parameter) of the attributes and choose the best split from among those variables. The classification of an unlabeled instance is performed using majority vote. Originally, the random forests algorithm applies only to building decision trees, and is not applicable to all types of classifiers, because it involves picking a different subset of the attributes in each node of the tree. Nevertheless, the main step of the random forest algorithm can be easily replaced with the

broader "random subspace method" [65], which can be applied to many other inducers, such as nearest neighbor classifiers [68] or linear discriminators [157].

One important advantage of the random forest method is its ability to handle a very large number of input attributes [157]. Another important feature of the random forest is that it is fast.

Using an extensive simulation study, Archer and Kimes [8] examine the effectiveness of Random Forest variable importance measures in identifying the true predictor among a large number of candidate predictors. They concluded that the Random Forest technique is useful in domains which require both an accurate classifier and insight regarding the discriminative ability of individual attribute (like in microarray studies).

5.2.4. Cross-validated Committees

This procedure creates k classifiers by partitioning the training set into k-equal-sized sets and training, in turn, on all but the i-th set. This method, first used by Gams [60], employed 10-fold partitioning. Parmanto *et al.* [118] have also used this idea for creating an ensemble of neural networks. Domingos [48] used cross-validated committees to speed up his own rule induction algorithm RISE, whose complexity is $O(n^2)$, making it unsuitable for processing large databases. In this case, partitioning is applied by predetermining a maximum number of examples to which the algorithm can be applied at once. The full training set is randomly divided into approximately equal-sized partitions. RISE is then run on each partition separately. Each set of rules grown from the examples in partition p is tested on the examples in partition $p+1$, in order to reduce overfitting and to improve accuracy.

6. Ensemble Diversity

In an ensemble, the combination of the output of several classifiers is only useful if they disagree about some inputs [171].

Creating an ensemble in which each classifier is as different as possible while still being consistent with the training set is theoretically known to be an important feature for obtaining improved ensemble performance [85]. According to [71], diversified classifiers lead to uncorrelated errors, which in turn improve classification accuracy.

In the regression context, the bias-variance-covariance decomposition has been suggested to explain why and how diversity between individual models contributes toward overall ensemble accuracy. Nevertheless, in the classification context, there is no complete and agreed upon theory [25]. More specifically, there is no simple analogue of variance-covariance decomposition for the zero-one loss function. Instead, there are several ways to define this decomposition. Each way has its own assumptions.

Sharkey [154] suggested a taxonomy of methods for creating diversity in ensembles of neural networks. More specifically, Sharkey's taxonomy refers to four different aspects: the initial weights; the training data used; the architecture of the networks; and the training algorithm used.

Brown et al. [25] suggest a different taxonomy which consists of the following branches: varying the starting points within the hypothesis space; varying the set of hypotheses that are accessible by the ensemble members (for instance by manipulating the training set); and varying the way each member traverses the hypothesis space.

In this paper we suggest the following taxonomy. Note however that the components of this taxonomy are not mutually exclusive, namely, there are a few algorithms which combine two of them.

1. Manipulating the Inducer – We manipulate the way in which the base inducer is used. More specifically each ensemble member is trained with an inducer that is differently manipulated.
2. Manipulating the Training Sample – We vary the input that is used by the inducer for training. Each member is trained from a different training set.
3. Changing the target attribute representation – Each classifier in the ensemble solve a different target concept.
4. Partitioning the hypothesis space – Each member is trained on a different hypothesis subspace.
5. Hybridization – Diversity is obtained by using various base inducers or ensemble strategies.

6.1. Manipulating the Inducer

A simple method for gaining diversity is to manipulate the inducer used for creating the classifiers. Below we survey several strategies to gain this diversity.

6.1.1. Manipulation of the inducer's parameters

The base inducer usually can be controlled by a set of parameters. For example, the well known decision tree inducer C4.5 has the confidence level parameter that greatly affects learning. Drucker [49] examine the effect of early pruning of decision trees on the performance of the entire ensemble. When an algorithm (such as decision tree) is used as a single strong learner, then certain aspects should be taken into consideration. But when the same algorithm is used as a weak learner then other aspects should be taken into consideration.

In the neural network community, there were several attempts to gain diversity by using different number of nodes [119, 186]. Nevertheless, these researches conclude that variation in numbers of hidden nodes is not effective method of creating diversity in neural network ensembles. Nevertheless the CNNE algorithm [76] which simultaneously determines the ensemble size along with the number of hidden nodes in individual NNs, has shown encouraging results.

Another effective approach for ANNs is to use several network topologies. For instance the Addemup algorithm [117] uses genetic algorithm to select the network topologies composing the ensemble. Addemup trains with standard backpropagation, and then selects groups of networks with a good error diversity according to the measurement of diversity.

6.1.2. Starting Point in Hypothesis Space

Some inducers can gain diversity by starting the search in the Hypothesis Space from different points. For example the simplest way to manipulate the back-propagation inducer is to assign different initial weights to the network [84]. Experimental study indicates that the resulting networks differed in the number of cycles in which they took to converge upon a solution, and in whether they converged at all. While it is very simple way to gain diversity, it is now generally accepted that it is not sufficient for achieving good diversity [25].

6.1.3. Hypothesis Space Traversal

These techniques alter the way the inducer traverses the space, thereby leading different classifiers to converge to different hypotheses [25]. We differentiate between two techniques for manipulating the space traversal for gaining diversity: Random and Collective-Performance.

Random-based strategy

The idea in this case is to "inject randomness" into the inducers in order to increase the independence among the ensemble's members. Ali and Pazzani [4] propose to change the rule learning HYDRA algorithm in the following way: Instead of selecting the best attribute at each stage (using, for instance, an information gain measure), the attribute is selected randomly such that its probability of being selected is proportional to its measured value. A similar idea has been implemented for C4.5 decision trees [45]. Instead of selecting the best attribute in each stage, it selects randomly (with equal probability) an attribute from the set of the best 20 attributes.

Collective-Performance-based strategy

In this case the evaluation function used in the induction of each member is extended to include a penalty term that encourages diversity. The most studied penalty method is the Negative Correlation Learning [24, 144]. The idea of negative correlation learning is to encourage different individual classifiers in the ensemble to represent different subspaces of the problem. While simultaneously creating the classifiers, the classifiers may interact with each other in order to specialize (for instance by using a correlation penalty term in the error function to encourage such specialization).

6.2. Manipulating the training samples

In this method, each classifier is trained on a different variation or subset of the original dataset. This method is useful for inducers whose variance-error factor is relatively large (such as decision trees and neural networks). That is to say, small changes in the training set may cause a major change in the obtained classifier. This category contains procedures such as bagging, boosting and cross-validated committees.

6.2.1. Resampling

The distribution of instances among the different classifier could be random as in the bagging algorithm or in the arbiter trees. Other methods distribute the instances based on the class distribution such that the class distribution in each subset is approximately the same as that in the entire dataset. It has been shown that proportional distribution as used in combiner trees [30] can achieve higher accuracy than random distribution.

Instead of perform sampling with replacement, some methods (like AdaBoost or Wagging) manipulate the weights that are attached to each instance in the training set. The base inducer should be capable to take these weights into account. Recently a novel framework was proposed in which each instance contributes to the committee formation with a fixed weight, while contributing with different individual weights to the derivation of the different constituent classifiers [33]. This approach encourages model diversity without biasing the ensemble inadvertently towards any particular instance.

6.2.2. Creation

The DECORATE algorithm [105] is a dependent approach in which the ensemble is generated iteratively, learning a classifier at each iteration and adding it to the current ensemble. The first member is created by using the base induction algorithm on the original training set. The successive classifiers are trained on an artificial set that combines instances from the original training set and also on some fabricated instances. In each iteration, the input attribute values of the fabricated instances are generated according to the original data distribution. On the other hand, the target values of these instances are selected so as to differ maximally from the current ensemble predictions. Comprehensive experiments have demonstrated that this technique is consistently more accurate than the base classifier, Bagging and Random Forests. Decorate also obtains higher accuracy than boosting on small training sets, and achieves comparable performance on larger training sets.

6.3. Manipulating the target attribute representation

In methods that manipulate the target attribute, instead of inducing a single complicated classifier, several classifiers with different and usually simpler representations of the target attribute are induced. This manipulation can be based on an aggregation of the original target's values (known as *Concept Aggregation*) or more complicated functions (known as *Function Decomposition*).

Classical concept aggregation replaces the original target attribute with a function, such that the domain of the new target attribute is smaller than the original one [27].

The idea to convert K class classification problems into K-two class classification problems has been proposed by [6]. Each problem considers the discrimination of one class to the other classes. Lu and Ito [99] extend Anand's method and propose a new method for manipulating the data based on the class relations among the training data. By using this method, they divide a

K class classification problem into a series of $K(K-1)/2$ two-class problems where each problem considers the discrimination of one class to each one of the other classes. The researchers used neural networks to examine this idea.

A general concept aggregation algorithm called *Error-Correcting Output Coding* (ECOC) uses a code matrix to decompose a multi-class problem into multiple binary problems [44]. ECOC for multi-class classification hinges on the design of the code matrix.

Data-driven Error Correcting Output Coding (DECOC) [194]. explore the distribution of data classes and optimize both the composition and the number of base learners to design an effective and compact code matrix. Specifically, DECOC calculate the confidence score of each base classifier based on the structural information of the training data and use sorted confidence scores to assist the determination of code matrix of ECOC. The results show that the proposed DECOC is able to deliver competitive accuracy compared with other ECOC methods, using parsimonious base learners than the pairwise coupling (one-vs-one) decomposition scheme.

Sivalingam et al. [160] propose to transform a multiclasses recognition problem into a minimal binary classification problem using the Minimal Classification Method (MCM) aided with error correcting codes. The MCM requires only $log_2 K$ classifications because instead of separating only two classes at each classification, this method separate two groups of multiple classes. Thus the MCM requires small number of classifiers and still provide similar accuracy performance.

A general-purpose function decomposition approach for machine learning was proposed in [192]. According to this approach, attributes are transformed into new concepts in an iterative manner to create a hierarchy of concepts.

6.4. Partitioning

Partitioning means dividing the original training set into smaller training sets. A different classifier is trained on each sub-sample. After all classifiers are constructed, the models are combined in some fashion [101]. There are two obvious ways to partition the original dataset: Horizontal Partitioning and Vertical Partitioning. In horizontal partitioning the original dataset is partitioned into several datasets that have the same features as the original dataset, each containing a subset of the instances in the original. In vertical partitioning the original dataset is partitioned into several datasets that have the same number of instances as the original dataset, each containing a subset of the original set of features.

In order to illustrate the idea of partitioning, consider the training set in Table 1 which contains a segment of the Iris dataset. This is one of the best known datasets in the pattern recognition literature. The goal in this case is to classify flowers into the Iris subgeni according to their characteristic features. The dataset contains three classes that correspond to three types of iris flowers: $dom\,(y) = \{IrisSetosa, IrisVersicolor, IrisVirginica\}$. Each pattern is characterized by four numeric features (measured in centimeters):

$A = \{sepallength, sepalwidth, petallength, petalwidth\}$. Tables 2 and 3 respectively illustrate mutually exclusive horizontal and vertical partitions of the Iris dataset. Note that despite the mutually exclusiveness, the class attribute must be included in each vertical partition.

Table 1: The Iris Dataset consisting of Four Numeric Features and Three Possible Classes.

Sepal Length	Sepal Width	Petal Length	Petal Width	Class (Iris Type)
5.1	3.5	1.4	0.2	Iris-setosa
4.9	3.0	1.4	0.2	Iris-setosa
6.0	2.7	5.1	1.6	Iris-versicolor
5.8	2.7	5.1	1.9	Iris-virginica
5.0	3.3	1.4	0.2	Iris-setosa
5.7	2.8	4.5	1.3	Iris-versicolor
5.1	3.8	1.6	0.2	Iris-setosa

Table 2: Horizontal Partitioning of the Iris Dataset.

Sepal Length	Sepal Width	Petal Length	Petal Width	Class (Iris Type)
5.1	3.5	1.4	0.2	Iris-setosa
4.9	3.0	1.4	0.2	Iris-setosa
6.0	2.7	5.1	1.6	Iris-versicolor

Sepal Length	Sepal Width	Petal Length	Petal Width	Class (Iris Type)
5.8	2.7	5.1	1.9	Iris-virginica
5.0	3.3	1.4	0.2	Iris-setosa
5.7	2.8	4.5	1.3	Iris-versicolor
5.1	3.8	1.6	0.2	Iris-setosa

6.4.1. Horizontal Partitioning

Some argue that classic ensemble techniques (such as boosting and bagging) have limitations on massive datasets, because the size of the dataset can become a bottleneck [32]. Moreover, it is suggested that partitioning the datasets into random, disjoint partitions will not only overcome the issue of exceeding memory size, but will also lead to creating an ensemble of diverse and accurate classifiers, each built from a disjoint partition but with the aggregate processing all of the data. This can improve performance in a way that might not be possible by subsampling. More recently a framework for building thousands of classifiers that are trained from small subsets of data in a distributed environment was proposed [32]. The robust learning from bites (RLB) algorithm that was proposed by Christmann et al. [34] is also designed to work with large data sets.

Table 3: Vertical Partitioning of the Iris Dataset.

Petal Length	Petal Width	Class (Iris Type)
1.4	0.2	Iris-setosa
1.4	0.2	Iris-setosa
5.1	1.6	Iris-versicolor
5.1	1.9	Iris-virginica
1.4	0.2	Iris-setosa
4.5	1.3	Iris-versicolor
1.6	0.2	Iris-setosa

Sepal Length	Sepal Width	Class (Iris Type)
5.1	3.5	Iris-setosa
4.9	3.0	Iris-setosa
6.0	2.7	Iris-versicolor
5.8	2.7	Iris-virginica
5.0	3.3	Iris-setosa
5.7	2.8	Iris-versicolor
5.1	3.8	Iris-setosa

The CBCD (cluster-based concurrent decomposition) algorithm [139] first clusters the input space by using the K-means clustering algorithm. Then, it creates disjoint sub-samples using the clusters in such a way that each sub-sample is comprised of instances from all clusters and hence represents the entire dataset. An inducer is applied in turn to each sub-sample. A voting mechanism is used to combine the classifiers classifications. Experimental study indicates that the CBCD algorithm outperforms the bagging algorithm.

Up to this point, each classifier in the ensemble considers the entire input space. Alternatively, the original input space can be divided into several sub-spaces and each member in the ensemble explores only a certain sub-space. When using the late approach, one should decide if the subspaces will overlap. At one extreme, the original problem is decomposed into several mutually exclusive sub-problems, such that each subproblem is solved using a dedicated classifier. In such cases, the classifiers may have significant variations in their overall performance over different parts of the input space [172]. At the other extreme, each classifier solves the same original task. In such cases, "If the individual classifiers are then appropriately chosen and trained properly, their performances will be (relatively) comparable in any region of the problem space. [172]". However, usually the sub-spaces may have soft boundaries, namely sub-spaces are allowed to overlap.

Denison et al. [41] examine two schemas for partitioning the input space into disjoint subspaces: The BPM (Bayesian partition model) schema has been shown to be unsuitable when the training set is large or there are many input attributes. The PPM (product partition model) schema provides good results

in several cases especially in datasets where there are many irrelevant input attributes and it is less suitable to situations where there are strong interactions among input attributes.

Tao et al. [164] partition the space to improve the performance of content-based image retrieval (CBIR) based on the assumption that all positive instances are included in a set and the negative instances split into a small number of clusters. Thus, they first group the negative instances into clusters, and then train a set of classifiers between these negative clusters. Additional classifier is built for the single positive cluster; Finally, these classifiers are combined to make a single classifier.

In the neural-networks community, Nowlan and Hinton [114] examined the mixture of experts (ME) approach, which partitions the input space into several subspaces and assigns different experts (classifiers) to the different subspaces. The subspaces, in ME, have soft boundaries (i.e., they are allowed to overlap). A gating network then combines the experts' outputs and produces a composite decision. An extension to the basic mixture of experts, known as hierarchical mixtures of experts (HME), has been proposed in [79]. This extension decomposes the space into sub-spaces, and then recursively decomposes each sub-space into sub-spaces.

Some researchers have used clustering techniques to partition the space [140, 133]. The basic idea is to partition the input space into mutually exclusive subsets using K-means clustering algorithm. An analysis of the results shows that the proposed method is well suited for datasets of numeric input attributes and that its performance is influenced by the dataset size and its homogeneity.

NBTree [83] induces a decision tree and a Naïve Bayes hybrid classifier. To induce an NBTree, the input space is recursively partitioned according to attributes values. The result of the recursive partitioning is a decision tree whose terminal nodes are Naïve Bayes classifiers. Since subjecting a terminal node to a Naïve Bayes classifier means that the hybrid classifier may classify two instances from a single hyper-rectangle region into distinct classes, the NBTree is more flexible than a pure decision tree. More recently Cohen et al. [36] generalizes the NBTree idea and examines a decision-tree framework for space decomposition. According to this framework, the original instance-space is hierarchically partitioned into multiple subspaces and a distinct classifier (such as neural network) is assigned to each subspace. Subsequently, an unlabeled, previously-unseen instance is classified by employing the classifier that was assigned to the subspace to which the instance belongs.

Altincay [5] proposes the use of model ensemble-based nodes where a multitude of models are considered for making decisions at each node. The ensemble members are generated by perturbing the model parameters and input attributes. In generating model ensembles multi-layer perceptron (MLP), linear multivariate perceptron and Fishers linear discriminant type models are considered. One of the main strengths of the proposed approach is that it uses small number of training samples that reach at nodes close to the leaf in an efficient way. Experiments conducted on several datasets and three model types indicate that the proposed approach achieves better classification accuracies compared to

individual nodes, even in cases when only one model class is used in generating ensemble members.

6.4.2. Vertical (Feature set) partitioning

In this section we describe ensemble methods that vertically partition the original training dataset for creating the ensemble members. The idea is to simply give each classifier a different projection of the training set. Tumer and Oza. [173] claim that feature set partitioning potentially facilitate the creation of a classifier for high dimensionality data sets without the feature selection drawbacks. Moreover, these methods can be used to improve the classification performance due to the reduced correlation among the classifiers. Bryll *et al.* [23] also indicate that the reduced size of the dataset implies faster induction of classifiers. Feature set partitioning avoids the class under-representation which may happen in instance subsets methods such as bagging. There are three popular strategies for creating feature subset-based ensembles: random-based, reduct-based and collective-performance-based strategy.

Random-based strategy The most straightforward techniques for creating feature subset-based ensemble are based on random selection or random projection [147]. Ho [65] uses random subsets to create forest of decision trees. The ensemble is constructed systematically by pseudo-randomly selecting subsets of features. The training instances are projected to each subset and a decision tree is constructed using the projected training samples. The process is repeated several times to create the forest. The classifications of the individual trees are combined by averaging the conditional probability of each class at the leaves (distribution summation). Ho shows that simple random selection of feature subsets may be an effective technique because the diversity of the ensemble members compensates for their lack of accuracy.

Tao and Tang [165] propose a relevance feedback algorithm using the Random Subspace Method to overcome the SVM's unstable and over-fitting problems that are common to content-based image retrieval. The idea is to build a strong classifier with a set of weak classifiers, which are trained on different randomly sampled features. Using this method, they construct a multiple number of SVMs with no over-fitting problem. Finally they combine these SVMs using the majority voting rule in attempt to solve the unstable problem.

Bay [15] proposed MFS which uses simple voting in order to combine outputs from multiple KNN (K-Nearest Neighbor) classifiers, each having access only to a random subset of the original features. Each classifier employs the same number of features. This procedure resembles the random subspaces methods.

Bryll *et al.* [23] introduce attribute bagging (AB) which combine random subsets of features. AB first finds an appropriate subset size by a random search in the feature subset dimensionality. It then randomly selects subsets of features, creating projections of the training set on which the classifiers are trained. A technique for building ensembles of simple Bayesian classifiers in random feature subsets was also examined [168] for improving medical applications.

Reduct-based strategy

A reduct is defined as the smallest feature subset which has the same predictive power as the whole feature set. By definition, the size of the ensembles that were created using reducts are limited to the number of features. There have been several attempts to create classifier ensembles by combining several reducts. Wu *et al.* [184] introduce the worst-attribute-drop-first algorithm to find a set of significant reducts and then combine them using naïve Bayes. Bao and Ishii [12] examine the idea of combining multiple K-nearest neighbor classifiers for text classification by reducts. Hu *et al.* [74] propose several techniques to construct decision forests, in which every tree is built on a different reduct. The classifications of the various trees are combined using a voting mechanism.

Collective-Performance-based strategy

Cunningham and Carney [38] introduced an ensemble feature selection strategy that randomly constructs the initial ensemble. Then, an iterative refinement is performed based on a hill-climbing search in order to improve the accuracy and diversity of the base classifiers. For all the feature subsets, an attempt is made to switch (include or delete) each feature. If the resulting feature subset produces a better performance on the validation set, that change is kept. This process is continued until no further improvements are obtained. Similarly, Zenobi and Cunningham [187] suggest that the search for the different feature subsets will not be solely guided by the associated error but also by the disagreement among the ensemble members.

Tumer and Oza [173] present a new method called input decimation (ID), which selects feature subsets based on the correlations between individual features and class labels. This experimental study shows that ID can outperform simple random selection of feature subsets.

Tsymbal *et al.* [169] compare several feature selection methods that incorporate diversity as a component of the fitness function in the search for the best collection of feature subsets. This study shows that there are some datasets in which the ensemble feature selection method can be sensitive to the choice of the diversity measure. Moreover, no particular measure is superior in all cases.

Gunter and Bunke [62] suggest employing a feature subset search algorithm in order to find different subsets of the given features. The feature subset search algorithm not only takes the performance of the ensemble into account, but also directly supports diversity of subsets of features.

Combining genetic search with ensemble feature selection was also examined in the literature. Opitz and Shavlik [117] applied GAs to ensembles using genetic operators that were designed explicitly for hidden nodes in knowledge-based neural networks. In a later research, Opitz [115] used genetic search for ensemble feature selection. This genetic ensemble feature selection (GEFS) strategy begins by creating an initial population of classifiers where each classifier is generated by randomly selecting a different subset of features. Then, new candidate classifiers are continually produced by using the genetic operators of crossover and mutation on the feature subsets. The final ensemble is composed of the most fitted classifiers. Similarly, the genetic algorithm that Hu *et al.* [74]

use for selecting the reducts to be included in the final ensemble, first creates N reducts then it trains N decision trees using these reducts. It finally uses a GA for selecting which of the N decision trees are included in the final forest.

Disjoint feature set partitioning In this section we discuss ensembles in which the subsets are pairwise disjoint subsets. Thus, a set of classifiers is trained such that each classifier employs a different subset of the original feature set. Subsequently, an unlabelled instance is classified by combining the classifications of all classifiers.

Several researchers have shown that this disjoint partitioning methodology can be appropriate for classification tasks with a large number of features [132, 89, 100, 135]. Specifically, Ahn et al. [3] indicate that random partition of the input attribute set into several subsets such that each classifier is induced from a different subset, is particularly useful for high-dimensional datasets. Their experiments indicate that for unbalanced data, their approach maintains the balance between sensitivity and specificity more adequately by adjusting the decision threshold in training phase.

In one research, the features are grouped according to the feature type: nominal value features, numeric value features and text value features [89]. A similar approach was also used for developing the linear Bayes classifier [59]. The basic idea consists of aggregating the features into two subsets: the first subset containing only the nominal features and the second only the continuous features.

In another research, the feature set was decomposed according to the target class [171]. For each class, the features with low correlation relating to that class were removed. This method was applied on a feature set of 25 sonar signals where the target was to identify the meaning of the sound (whale, cracking ice, etc.).

The feature set decomposition can be obtained by grouping features based on pairwise mutual information, with statistically similar features assigned to the same group [94]. For this purpose one can use an existing hierarchical clustering algorithm. As a consequence, several feature subsets are constructed by selecting one feature from each group. A neural network is subsequently constructed for each subset. All networks are then combined.

In statistics literature, the well-known feature-oriented ensemble algorithm is the MARS algorithm [56]. In this algorithm, a multiple regression function is approximated using linear splines and their tensor products. It has been shown that the algorithm performs an ANOVA decomposition, namely, the regression function is represented as a grand total of several sums. The first sum is of all basic functions that involve only a single attribute. The second sum is of all basic functions that involve exactly two attributes, representing (if present) two-variable interactions. Similarly, the third sum represents (if present) the contributions from three-variable interactions, and so on. In a recent study, severl methods for combining different feature selection results have been proposed [137]. The experimental results indicate that combining different feature selection methods can significantly improve the accuracy results.

A general framework that searches for helpful feature set disjoints partitioning structures has also been proposed [138]. This framework nests many algorithms, two of which are tested empirically over a set of benchmark datasets. This work indicates that feature set decomposition can increase the accuracy of decision trees. More recently, genetic algorithm has been successfully applied for feature set partitioning [129]. This GA uses a new encoding schema and a Vapnik–Chervonenkis dimension bound for evaluating the fitness function. The algorithm also suggests a new caching mechanism to speed up the execution and avoid recreation of the same classifiers.

6.5. Multi-Inducers

In Multi-Inducer strategy, diversity is obtained by using different types of inducers [111]. Each inducer contains an explicit or implicit bias that leads it to prefer certain generalizations over others. Ideally, this multi-inducer strategy would always perform as well as the best of its ingredients. Even more ambitiously, there is hope that this combination of paradigms might produce synergistic effects, leading to levels of accuracy that neither atomic approach by itself would be able to achieve.

Most research in this area has been concerned with combining empirical approaches with analytical methods (see for instance [166]. Woods et al. [182] combine four types of base inducers (decision trees, neural networks, k-nearest neighbors and quadratic Bayes). They then estimate local accuracy in the feature space to choose the appropriate classifier for a given new unlabeled instance. Wang et al. [178] examined the usefulness of adding decision trees to an ensemble of neural networks. The researchers concluded that adding a few decision trees (but not too many) usually improved the performance. Langdon et al. [91] proposed using Genetic Programming to find an appropriate rule for combining decision trees with neural networks.

Brodley [22] proposed the model class selection (MCS) system. MCS fits different classifiers to different subspaces of the input space, by employing one of three classification methods (a decision-tree, a discriminant function or an instance-based method). In order to select the classification method, MCS uses the characteristics of the underlined training-set, and a collection of expert rules. Brodley's expert-rules were based on empirical comparisons of the methods' performance (i.e., on prior knowledge).

The NeC4.5 algorithm, which integrates decision tree with neural networks [193], first trains a neural network ensemble. Then, the trained ensemble is employed to generate a new training set by replacing the desired class labels of the original training examples with the output from the trained ensemble. Some extra training examples are also generated from the trained ensemble and added to the new training set. Finally, a C4.5 decision tree is grown from the new training set. Since its learning results are decision trees, the comprehensibility of NeC4.5 is better than that of neural network ensembles.

Using several inducers can solve the dilemma which arises from the "no free lunch" theorem. This theorem implies that a certain inducer will be successful only insofar its bias matches the characteristics of the application domain [17].

Thus, given a certain application, the practitioner need to decide which inducer should be used. Using the multi-inducer obviate the need to try each one and simplifying the entire process.

6.6. Measuring the Diversity

As stated above, it is usually assumed that increasing diversity may decrease ensemble error [187]. For regression problems, *variance* is usually used to measure diversity [85]. In such cases it can be easily shown that the ensemble error can be reduced by increasing ensemble diversity while maintaining the average error of a single model.

In classification problems, a more complicated measure is required to evaluate the diversity. There have been several attempts to define diversity measure for classification tasks.

In the neural network literature two measures are presented for examining diversity:

- Classification coverage: An instance is covered by a classifier, if it yields a correct classification.
- Coincident errors: A coincident error amongst the classifiers occurs when more than one member misclassifies a given instance.

Based on these two measures, Sharkey [153] defined four diversity levels:

- Level 1 - No coincident errors and the classification function is completely covered by a majority vote of the members.
- Level 2 - Coincident errors may occur, but the classification function is completely covered by a majority vote.
- Level 3 - A majority vote will not always correctly classify a given instance, but at least one ensemble member always correctly classifies it.
- Level 4 - The function is not always covered by the members of the ensemble.

Brown et al. [25] claim that the above four-level scheme provides no indication of how typical the error behavior described by the assigned diversity level is. This claim, especially, holds when the ensemble exhibits different diversity levels on different subsets of input space.

There are other more quantitative measures which categorize these measures into two types [25]: pairwise and non-pairwise. Pairwise measures calculate the average of a particular distance metric between all possible pairings of members in the ensemble, such as Q-statistic [25] or kappa-statistic [104]. The non-pairwise measures either use the idea of entropy (such as [38]) or calculate a correlation of each ensemble member with the averaged output. The comparison of several measures of diversity has resulted in the conclusion that most of them are correlated [88].

Here we focus on the pairwise diversity measures. For an ensemble of n classifiers the total pairwise diversity measure is calculated as the mean pairwise measure over all $n \cdot (n-1)/2$ pairs of classifiers:

$$F_{Total} = \frac{2}{n(n-1)} \sum_{\forall i \neq j} f_{i,j} \tag{12}$$

where $f_{i,j}$ is a similarity or diversity measure of two classifiers outputs i and j. Kuncheva and Whitaker (2003) find the following two diversity pairwise measures useful:

1. The disagreement measure is defined as the ratio between the number of instances on which one classifier is correct and its counterpart is incorrect to the total number of instances:

$$Dis_{i,j} = \frac{m_{\bar{i}\,j} + m_{i\,\bar{j}}}{m_{\bar{i}\,j} + m_{i\,\bar{j}} + m_{i\,j} + m_{\bar{i}\,\bar{j}}} \tag{13}$$

   where $m_{i\,j}$ specifies the number of instances in which both classifier i and classifier j are correct while $m_{\bar{i}\,\bar{j}}$ indicates the number of instances that are misclassified by both classifiers. Similarly, $m_{i\,\bar{j}}$ and $m_{\bar{i}\,j}$ indicate the number of instances in which one classifier has correctly classified the instances but its counterpart has misclassified these instances.
2. The double-fault measure is defined as the proportion of the cases that have been misclassified by both classifiers:

$$DF_{i,j} = \frac{m_{\bar{i}\,\bar{j}}}{m_{\bar{i}\,j} + m_{i\,\bar{j}} + m_{i\,j} + m_{\bar{i}\,\bar{j}}} \tag{14}$$

Instead of measuring the diversity, we can complementarily use the following pairwise similarity measures:

1. The Q statistics is defined as:

$$Q_{i,j} = \left(m_{i\,j} \cdot m_{\bar{i}\,\bar{j}} - m_{i\,\bar{j}} \cdot m_{\bar{i}\,j}\right) / \left(m_{i\,j} \cdot m_{\bar{i}\,\bar{j}} + m_{i\,\bar{j}} \cdot m_{\bar{i}\,j}\right) \tag{15}$$

   The Q measures varies between -1 and 1, where positive values indicate that the two classifiers are correlated (namely they tend to correctly classify the same instances). A value close to 0 indicates that the classifiers are independent.
2. The correlation coefficient – The ρ measure is very similar to the Q measure. It has the same numerator as Q measure. Moreover, it always has the same sign but the value magnitude is never greater than the corresponding Q value:

$$\rho_{i,j} = \frac{\left(m_{i\,j} \cdot m_{\bar{i}\,\bar{j}} - m_{i\,\bar{j}} \cdot m_{\bar{i}\,j}\right)}{\sqrt{\left(m_{i\,j} + m_{i\,\bar{j}}\right) \cdot \left(m_{i\,j} + m_{\bar{i}\,j}\right) \cdot \left(m_{\bar{i}\,\bar{j}} + m_{i\,\bar{j}}\right) \cdot \left(m_{\bar{i}\,\bar{j}} + m_{\bar{i}\,j}\right)}} \tag{16}$$

Kuncheva and Whitaker [88] show that these measures are strongly correlated between themselves. Still on specific real classification tasks, the measures might behave differently, so they can be used as a complementary set. Nevertheless, Kuncheva and Whitaker [88] could not find a definitive connection between the measures and the improvement of the accuracy. Thus, they conclude that it is unclear if diversity measures have any practical value in building classifier ensembles.

7. Ensemble Size

7.1. Selecting the Ensemble Size

An important aspect of ensemble methods is to define how many component classifiers should be used. There are several factors that may determine this size:

- Desired accuracy — In most cases, ensembles containing ten classifiers are sufficient for reducing the error rate [64]. Nevertheless, there is empirical evidence indicating that: when AdaBoost uses decision trees, error reduction is observed in even relatively large ensembles containing 25 classifiers [116]. In disjoint partitioning approaches, there may be a trade-off between the number of subsets and the final accuracy. The size of each subset cannot be too small because sufficient data must be available for each learning process to produce an effective classifier.

- Computational cost — Increasing the number of classifiers usually increases computational cost and decreases their comprehensibility. For that reason, users may set their preferences by predefining the ensemble size limit.

- The nature of the classification problem - In some ensemble methods, the nature of the classification problem that is to be solved, determines the number of classifiers. For instance in the ECOC algorithm the number of classes determine the ensemble size.

- Number of processors available — In independent methods, the number of processors available for parallel learning could be put as an upper bound on the number of classifiers that are treated in paralleled process.

There are three approaches for determining the ensemble size, as described by the following subsections.

7.2. Pre selection of the ensemble size

This is the most simple way to determine the ensemble size. Many ensemble algorithms have a controlling parameter such as "number of iterations", which can be set by the user. Algorithms such as Bagging belong to this category. In other cases the nature of the classification problem determine the number of members (such as in the case of ECOC).

7.3. Selection of the ensemble size while training

There are ensemble algorithms that try to determine the best ensemble size while training. Usually as new classifiers are added to the ensemble these algorithms check if the contribution of the last classifier to the ensemble performance is still significant. If it is not, the ensemble algorithm stops. Usually these algorithms also have a controlling parameter which bounds the maximum size of the ensemble.

Random forests algorithm uses out-of-bag (oob) procedure to get an unbiased estimate of the test set error [20]. The effectiveness of using out-of-bag error estimate, to decide when a sufficient number of classification trees have been recently examined in [10]. Specifically, the algorithm works by first smoothing the out-of-bag error graph with a sliding window in order to reduce the variance. After the smoothing has been completed, the algorithm takes a larger window on the smoothed data points and determines the maximum accuracy within that window. It continues to process windows until the maximum accuracy within a particular window no longer increases. At this point, the stopping criterion has been reached and the algorithm returns the ensemble with the maximum raw accuracy from within that window. It has been shown that out-of-bag obtain an accurate ensemble for those methods that incorporate bagging into the construction of the ensemble.

7.4. Pruning - Post selection of the ensemble size

As in decision tree induction, it is sometimes useful to let the ensemble grow freely and then prune the ensemble in order to get more effective and compact ensembles. Post selection of the ensemble size allows ensemble optimization for such performance metrics as accuracy, cross entropy, mean precision, or the ROC area. Empirical examinations indicate that pruned ensembles may obtain a similar accuracy performance as the original ensemble [104]. In another empirical study that was conducted in order to understand the affect of ensemble sizes on ensemble accuracy and diversity, it has been shown that it is feasible to keep a small ensemble while maintaining accuracy and diversity similar to those of a full ensemble [97].

The pruning methods can be divided into two groups: pre-combining pruning methods and post-combining pruning methods.

7.4.1. Pre-combining pruning

Pre-combining pruning is performed before combining the classifiers. Classifiers that seem to perform well are included in the ensemble. Prodromidis et al. [122] present three methods for pre-combining pruning: based on an individual classification performance on a separate validation set, diversity metrics, the ability of classifiers to classify correctly specific classes.

In attribute bagging [23], classification accuracy of randomly selected m-attribute subsets is evaluated by using the wrapper approach and only the classifiers constructed on the highest ranking subsets participate in the ensemble voting.

7.4.2. post-combining pruning

In post-combining pruning methods, we remove classifiers based on their contribution to the collective.

Prodromidis [122] examines two methods for post-combining pruning assuming that the classifiers are combined using meta-combination method: Based on decision tree pruning and the correlation of the base classifier to the unpruned meta-classifier.

A forward stepwise selection procedure can be used in order to select the most relevant classifiers (that maximize the ensemble's performance) among thousands of classifiers [29]. It has been shown that for this purpose one can use feature selection algorithms. However, instead of selecting features one should select the ensemble's members [97].

Rokach et al. [136, 7] suggest first to rank the classifiers according to their ROC performance. Then, they suggest to plot a graph where the Y- axis displays a performance measure of the integrated classification . The X-axis presents the number of classifiers that participated in the combination. i.e., the first best classifiers from the list are combined by voting (assuming equal weights for now) with the rest getting zero weights. The ensemble size is chosen when there are several sequential points with no improvement.

The algorithm FS-PP-EROS generates a selective ensemble of rough subspaces [72]. The algorithm performs an accuracy-guided forward search and post-pruning strategy to select part of the base classifiers for constructing an efficient and effective ensemble system. The experimental results show that FS-PP-EROS outperform bagging and random subspace methods in terms of accuracy and size of ensemble systems.

The GASEN algorithm was developed for selecting the most appropriate classifiers in a given ensemble [191]. In the initialization phase, GASEN assigns a random weight to each of the classifiers. Consequently, it uses genetic algorithms to evolve those weights so that they can characterize to some extent the fitness of the classifiers in joining the ensemble. Finally, it removes from the ensemble those classifiers whose weight is less than a predefined threshold value.

Recently a revised version of the GASEN algorithm called GASEN-b has been suggested [190]. In this algorithm, instead of assigning a weight to each classifier, a bit is assigned to each classifier indicating whether it will be used in the final ensemble. In an experimental study the researchers showed that ensembles generated by a selective ensemble algorithm, which selects some of the trained C4.5 decision trees to make up an ensemble, may be not only smaller in size but also stronger in the generalization than ensembles generated by non-selective algorithms.

A study had compared several post combining pruning methods that were applied to Boosting and Bagging [181]. Specifically the following pruning methods have been compared: Minimum Error Pruning (MEP), Error-based Pruning (EBP), Reduced-Error Pruning(REP), Critical Value Pruning (CVP) and Cost-Complexity Pruning (CCP). The results indicate that if a single pruning method needs to be selected then overall the popular EBP makes a good choice.

A comparative study of pre combining pruning and post combining pruning methods when meta-combining methods are used has been performed in [122]. The results indicate that the post-combining pruning methods tend to perform better in this case.

Zhang et al. [189] use boosting for determining the order in which the base classifiers are fused, and then construct a pruned ensemble by stopping the fusion process early. Two heuristics rules are used to stop fusion: one is to select the upper twenty percent of the base classifiers from the ordered full Double-Bagging ensemble and the other is to stop the fusion when the weighted training error reaches 0.5.

Croux et al. [37] propose the idea of trimmed bagging which aims to prune classifiers that yield the highest error rates, as estimated by the out-of-bag error rate. It has been shown that trimmed bagging performs comparably to standard bagging when applied to unstable classifiers as decision trees, but yields improved accuracy when applied to more stable base classifiers, like support vector machines.

Rokach [134] introduces the Collective-Agreement-based Pruning (CAP) measure for selecting the most relevant classifiers in the ensemble. Rather than ranking individual members, CAP ranks subsets by considering the individual predictive ability of each member along with the degree of redundancy among them. Subsets whose members highly agree with the class while having low inter-agreement are preferred.

8. Cross-Inducer

This property indicates the relation between the ensemble technique and the inducer used. Some implementations are considered as an inducer-dependent type, namely these ensemble generators which use intrinsic inducer, have been developed specifically for a certain inducer. They can neither work nor guarantee effectiveness in any other induction method. For instance, the works of [64, 99, 152] were developed specifically for neural networks. Other procedures were developed specifically for SVM [163], decision trees [21, 138] and logistic regression [151].

Other implementations are considered to be the inducer-independent type. These implementations can be performed on any given inducer and are not limited to a specific inducer like the inducer-dependent.

9. Multistrategy Ensemble Learning

Multistrategy ensemble learning combines several ensemble strategies. It has been shown that this hybrid approach increases the diversity of ensemble members.

MultiBoosting, an extension to AdaBoost expressed by adding wagging-like features [179], can harness both AdaBoost's high bias and variance reduction with wagging's superior variance reduction. Using C4.5 as the base learning

algorithm, MultiBoosting, significantly more often than the reverse, produces decision committees with lower error than either AdaBoost or wagging . It also offers the further advantage over AdaBoost of suiting parallel execution. MultiBoosting has been further extended by adding the stochastic attribute selection committee learning strategy to boosting and wagging [180]. The latter's research has shown that combining ensemble strategies would increase diversity at the cost of a small increase in individual test error resulting in a trade-off that reduced overall ensemble test error.

Tao et al. [163] propose to combine bagging-based SVM and random subspace SVM in order to improve the relevance feedback in image retrieval, particularly when the number of labeled positive feedback samples is small. Additionally, Li et al. [93] develop a new machine learning technique, multitraining SVM (MTSVM), which combines the merits of the co-training technique and a random sampling method in the feature space.

Another multistrategy method suggests to create the ensemble by decomposing the original classification problem into several smaller and more manageable sub-problems[132]. This multistrategy uses an elementary decomposition framework that consists of five different elementary decompositions: Concept Aggregation, Function, Sample, Space and Feature Set. The concept of elementary decomposition can be used to obtain a complicated decomposition by using the elementary decomposition concept recursively. Given a certain problem, the procedure selects the most appropriate elementary decomposition (if any) to that problem. A suitable decomposer then decomposes the problem and provides a set of sub-problems. A similar procedure is performed on each sub-problem until no beneficial decomposition is anticipated. The selection of the best elementary decomposition for a given problem is performed by using a meta-learning approach.

10. Illustration of new taxonomy

Figure 10 presents the complete taxonomy based on the description provided in previous sections. Table 4 illustrates the categorization of 10 popular ensemble algorithms. It is not surprising that AdaBoost and Arc-x4 have the same characterization, since both of them instances of arcing. Similarly, bagging and wagging also have the same characteristics, because bagging is a specific case of wagging. The categorization for multistrategy ensemble techniques is not mutually exclusive and these techniques belong to more than one leaf within the same dimension. For instance, the diversity of Random Forest is gained by a combination of training set manipulation with random traversal of hypothesis space.

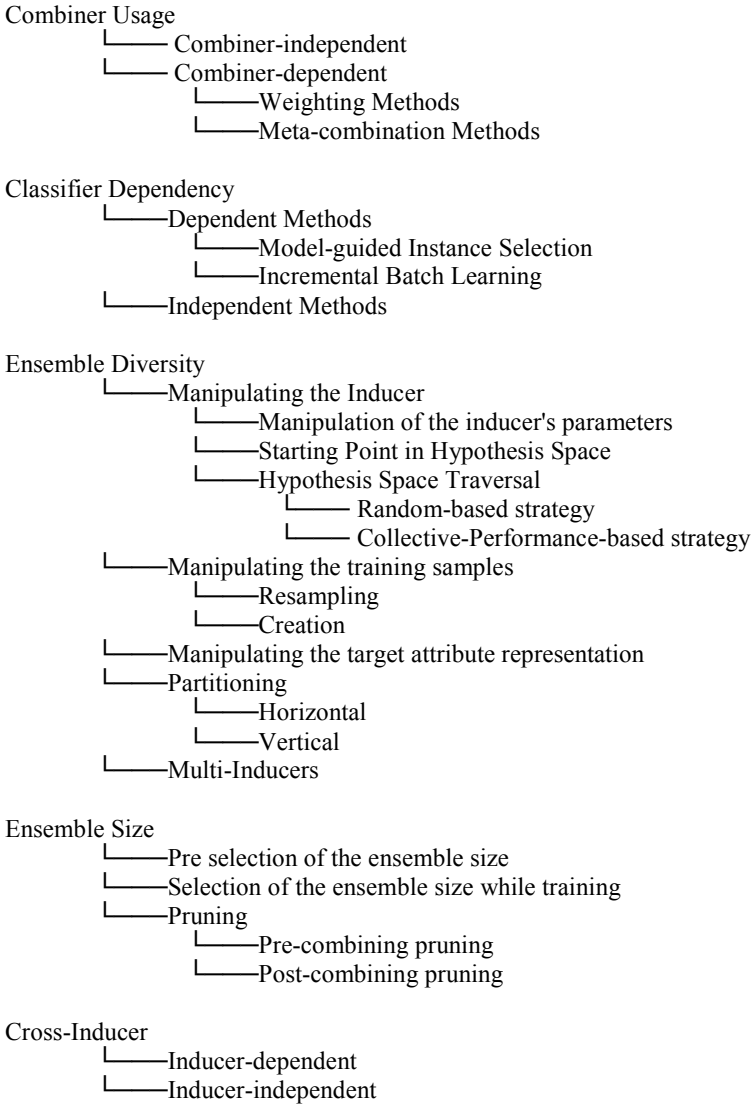

Figure 1: The complete ensemble taxonomy

Table 4: Illustration the categorization using the new taxonomy

Algorithm	Combiner Usage	Classifier Dependency	Ensemble Diversity	Ensemble Size	The Inducer Usage
AdaBoost [55]	Weighting	Model-guided	Resampling	Pre selection	Inducer-independent
Bagging [18]	Weighting	Independent	Resampling	Pre selection	Inducer-independent
RandomForest [21]	Weighting	Independent	Resampling + Random Traversal of Hypothesis Space	Pre selection	Inducer-dependent (Decision Tree)
Attribute Bagging [23]	Weighting	Vertical Partitioning	Pre-combining pruning	Inducer-independent	
DECORATE [105]	Weighting	Model-guided	Creation	Pre selection	Inducer-independent
Stacking [183]	Meta-combination	Independent	Multi-Inducers	Pre selection	Inducer-independent
ECOC [44]	Weighting	Independent	Manipulating target attribute	Pre selection	Inducer-independen
Arc-x4 [19]	Weighting	Model-guided	Resampling	Pre selection	Inducer-independent
MultiBoosting [179]	Weighting	Model-guided+ Independent	Resampling	Pre selection	Inducer-independent
Wagging [14]	Weighting	Independent	Resampling	Pre selection	Inducer-independent

11. Which Ensemble Method Should be Used?

The previous sections concentrated on design aspects of ensemble methods. Given the vast repertoire of ensemble methods to choose from, this section provides several selection criteria which aim to help practitioners selecting the most suitable ensemble method for their specific needs. For each criterion we also indicate how it can be evaluated.

11.1. Predictive performance

Historically predictive performance measures are the main criteria for selecting inducers. Moreover, the predictive performance measures, such as accuracy, are considered to be an objective and quantified, which can be easily used to benchmark algorithms.

In addition to the experimental studies that are performed to validate the contribution of a new specific ensemble method, there are several large comparative studies, which aim to assist the practitioner in his decision making.

Dietterich [45] has compared three methods for constructing forest of C4.5 classifiers: Randomizing, Bagging, and Boosting. The experiments show that when there is little noise in the data, boosting gives the best results. Bagging and Randomizing are usually equivalent. Another study [14] compared Bagging and Boosting using decision trees and naive Bayes. The study determines that Bagging reduced variance of unstable methods, while boosting methods reduced both the bias and variance of unstable methods but increased the variance for stable methods.

Additional study [116] that compared Bagging with Boosting using neural networks and decision trees indicates that Bagging is sometimes significantly less accurate than Boosting. The study indicates that the performance of the Boosting methods is much more sensitive to the characteristics of the dataset, specifically Boosting may overfit noisy data sets and reducing classification performance.

A recent research has experimentally evaluated bagging and seven other randomization-based approaches for creating an ensemble of decision tree classifiers [10]. Statistical tests were performed on experimental results from 57 publicly available datasets. When cross-validation comparisons were tested for statistical significance, the best method was statistically more accurate than bagging on only eight of the 57 datasets. Alternatively, examining the average ranks of the algorithms across the group of datasets, Banfield et al. found that boosting, random forests, and randomized trees is statistically significantly better than bagging.

Sohna and Shinb [159] compared the performance of several ensemble methods (bagging, modified random subspace method, classifier selection, parametric fusion) to a single classifier assuming that the base inducer is logistic regression. They argue that several factors should be taken into consideration when performing such comparison, including correlation between input variables; variance of observation, and training data set size. They show that for large training

sets, the performances of a single logistic regression and bagging are not significantly different. However, when training set size are small, bagging is superior to a single logistic regression classifier. When training data set size is small and correlation is strong, both modified random subspace method and bagging perform better than the other methods. When correlation is weak and variance is small, both parametric fusion and classifier selection algorithm appear to be the worst.

11.2. Computational Cost

It is important to check the method efficiency, i.e., does it produce results in a reasonable amount of time (i.e. complexity cost) relative to the data size, the nature of the base inducer and other variables. We will distinguish between training time and classification time:

- Training time - indicate the complexity cost for obtaining the ensemble of classifiers.
- The classifying time, which span from receiving new data until the data is classified. The nature of the application determines if this measure is constrained, or not.

11.3. Scalability

Scalability indicates the method ability to scale to large data sets. There are ensemble methods (such as partitioning methods) that are more suitable to scale to large dataset than other.

Moreover independent methods are considered to be more scalable than dependent methods because the former case, classifiers can be trained in parallel.

11.4. Flexibility

Flexibility indicates the ability to use any inducer (inducer-independent), any combiner (Combiner-independent), provide a solution to variety of classification tasks (for example it is should not be limited to a binary classification task), a set of controlling parameters which enable the user to examine several variations of the ensemble techniques.

11.5. Usability

Machine learning is highly iterative process. Practitioners typically adjust algorithm's parameters to generate better classifiers. A good ensemble method should provide a set of controlling parameters that are comprehensive and can be easily tuned.

11.6. Interpretability of the resulting ensemble

Interpretability indicates the user ability to understand the ensemble results. This is especially important in applications in which the user is required to understand the system behavior or explain its classification. For example the revised version of AdaBoost presented in [57] is considered to provide interpretable descriptions of the aggregate decision rule.

Interpretability is usually a subjective criterion. Nevertheless, there are several quantitative measures and indicators that can help us in evaluating this criterion. For example:

- Compactness - measures the knowledge representation size efficiency. Obviously, results presented by a smaller size are easier to understand. In ensemble methods compactness can be measured by the ensemble size (number of classifiers) and the complexity of each classifier. According to Freund and Mason [54] Even for modest values of ensemble size, boosting of decision trees could result in a final combined classifier with thousands (or millions) of nodes which it is difficult to visualize.

- Base Inducer Used - The base inducer used by the ensemble can determine its interpretability. For example, decision trees are easier to understand than black-box methods such as neural networks.

11.7. Software Availability

Software Availability of an ensemble method indicates how many off-the-shelf software packages support this ensemble method. High Availability implies that the practitioner can move from one software to another, without the need to replace his ensemble method. Table 5 indicates the popularity (as measured by the number of citation in Google scholar in June, 2007) of the ten methods presented in Table 4 and the total availability of these methods in five open source packages: Weka [53], Orange [40], Tanagra [125], RapidMiner (formerly YALE)[75], OpenDT [11], Java-ML /citeAbeel and R programming environment [126]. The table indicates that high popularity is a necessary condition for high availability, but still there are popular methods with relatively low availability.

11.8. Why is it difficult to choose?

The difficulty in choosing the ensemble methods results from the fact that this is a MCDM (Multiple Criteria Decision Making) problem. There is a trade off relationship among the criteria. Some criteria can not be measured in commensurate units. Thus, in order to systematically choose the right method, the practitioner is encouraged to implement one of the MCDM solving technique such as AHP (Analytic Hierarchy Process).

Moreover, the context of the specific classification problem to be solved has tremendous effect on the results. In general, all comparative studies that have been performed in the literature and aim to compare the predictive performance,

Table 5: Software Availability of the Ensemble Method

Algorithm	Google Scholar (June 2009)	Software Availability	Reference
AdaBoost	2730	Weka, Orange, Tanagra, RapidMiner, R	[55]
Bagging	4907	All	[18]
RandomForest	1988	All	[21]
DECORATE	81	Weka	[105]
MultiBoosting	184	Weka, RapidMiner	[179]
Wagging	993	Weka, RapidMiner	[14]
Attribute Bagging	52	Weka	[23]
Stacking	1582	Weka, Tanagra, RapidMiner	[183]
ECOC	1007	Weka, RapidMiner	[44]
Arc-x4	644	Weka, Tanagra	[19]

show that the no-free-lunch theorem holds [25, 159], i.e. the best ensemble technique depends much on the particular training dataset. Thus, the current challenge is to automatically choose the best ensemble technique. There are two alternatives to achieve this goal:

- The wrapper approach – Given a certain dataset, use each ensemble method and select the one that appears to give the highest success rate. The main advantage of this approach is its ability to predict quite well the performance of each examined method. The main disadvantage of this method is it's prolonged processing time. For some inducers the induction times may be very long, particularly in large real-life datasets. Several researchers have implemented this approach for selecting induction algorithms or dimension reduction algorithms and showed that it produces superior results [145].

- The meta-learning approach [176]– Based on datasets characteristics, the meta-classifier decides whether to use ensemble method or not and what technique to use. If a certain ensemble method outperforms other methods in a particular dataset, then one should expect that this method will be preferable when other problems with similar characteristics are presented. For this purpose one can employ meta-learning. Meta-learning is concerned with accumulating experience on the performance of multiple applications of a learning system. One possible output of the meta-learning process is a meta-classifier that is capable to indicate which learning method is most appropriate to a given problem. This goal can be accomplished by performing the following phases: In the first phase one

should examine the performance of all investigated ensemble methods on various datasets. Upon examination of each dataset, the characteristics of the dataset are extracted. The dataset's characteristics, together with the indication of the most preferable ensemble method, (in this dataset) are stored in a meta-dataset. This meta-dataset reflects the experience accumulated across different datasets. In the second phase, an inducer can be applied to this meta-dataset to induce a meta-classifier that can map a dataset to the most appropriate ensemble method (based on the characteristics of the dataset). In the last phase, the meta-classifier is actually used to match a new unseen dataset to the most appropriate ensemble method. Several researchers have implemented this approach for selecting an ensemble method and showed that it produces superior results [132]

12. Conclusion and Open Issues

This paper presented an updated taxonomy survey of ensemble methods for classification problems. It has been shown that most algorithms fit into a relatively simple taxonomy. We also presented several criteria for practically selecting an ensemble method.

There are several open issues in the field of ensemble methodology:

- In this paper we have proposed using five dimensions for ensemble taxonomy. It is natural that not all of them can be put into one framework. Other dimensions which are not considered in the proposed taxonomy can be used to categorize ensemble methods. For example it has been proposed to categorize combinations methods into three levels [185]: abstract, rank and measurement. In the abstract level, a classifier only outputs a single label. In the second level, a classifier ranks all labels or a subset of the labels in a queue. In the measurement level, a classifier attributes to each class a measurement value that reflects the degree of confidence that a specific input belongs to a given class.

- Experimental studies show that the success of an ensemble technique depends upon many factors, including the training sample size; the choice of a base classifier; the exact way in which the training set is modified; the choice of the combination method; and, finally, on the data distribution and the potential ability of the chosen base classifier to solve the problem. Thus it is necessary to develop universal criteria to predict the usefulness of an ensemble technique given a certain dataset.

- A closely related issue is explaining ensemble behavior and performance for generalized loss functions. Recently researchers have started to examine the behavior of various ensemble techniques. For instance, Rudin et al. [143] analyze AdaBoosts asymptotic behavior and describe many aspects of AdaBoosts dynamical traits including the fact that AdaBoost does not necessarily converge to a maximum margin solution. There are many interesting questions that still remain open.

- The relation between ensemble diversity to the ensemble performance has been formalized for regression problems; nevertheless, this has not yet been formalized for classification problems [25].
- Most of the ensemble learning research has concentrated on simple classification tasks. Further study is required to better apply this methodology to increasing variety of data types such as time series, spatial, multimedia, etc or derived supervised tasks such as active learning, sequence classification.

References

[1] Thomas Abeel, Yves Van de Peer, Yvan Saeys, Java-ML: A Machine Learning Library, Journal of Machine Learning Research 10 (2009) 931-934

[2] Adem, J., Gochet, W., 2004. Aggregating classifiers with mathematical programming. Comput. Statist. Data Anal. 47 (4), 791-807.

[3] Ahn H., Moon H., Fazzari M. J., Noha Lim,James J. Chen, Ralph L. Kodell, Classification by ensembles from random partitions of high-dimensional data, Computational Statistics and Data Analysis 51 (2007) 6166-6179

[4] Ali K. M., Pazzani M. J., Error Reduction through Learning Multiple Descriptions, Machine Learning, 24: 3, 173-202, 1996.

[5] Altincay H., Decision trees using model ensemble-based nodesPattern Recognition 40 (2007) 3540 - 3551

[6] Anand R, Methrotra K, Mohan CK, Ranka S. Efficient classification for multiclass problems using modular neural networks. IEEE Trans Neural Networks, 6(1): 117-125, 1995.

[7] Arbel, R. and Rokach, L., Classifier evaluation under limited resources, Pattern Recognition Letters, 27(14): 1619–1631, 2006, Elsevier.

[8] Archer K. J., Kimes R. V., Empirical characterization of random forest variable importance measures, Computational Statistics and Data Analysis 52 (2008) 2249-2260

[9] Averbuch, M. and Karson, T. and Ben-Ami, B. and Maimon, O. and Rokach, L., Context-sensitive medical information retrieval, The 11th World Congress on Medical Informatics (MEDINFO 2004), San Francisco, CA, September 2004, IOS Press, pp. 282-286

[10] Robert E. Banfield, Lawrence O. Hall, Kevin W. Bowyer, W.P. Kegelmeyer, A Comparison of Decision Tree Ensemble Creation Techniques,IEEE Transactions on Pattern Analysis and Machine Intelligence,vol. 29, no. 1, pp. 173-180, Jan., 2007 1.

[11] R. Banfield, OpenDT, http://opendt.sourceforge.net/, 2005

[12] Y Bao, N. Ishii, Combining multiple K-nearest neighbor classifiers for text classification by reducts. In: Proceedings of 5th international conference on discovery science, LNCS 2534, Springer, 2002, pp 340-347

[13] Bartlett P. and Shawe-Taylor J., Generalization Performance of Support Vector Machines and Other Pattern Classifiers, In "Advances in Kernel Methods, Support Vector Learning", Bernhard Scholkopf, Christopher J. C. Burges, and Alexander J. Smola (eds.), MIT Press, Cambridge, USA, 1998.

[14] Bauer, E. and Kohavi, R., "An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants". Machine Learning, 35: 1-38, 1999.

[15] Bay, S., Nearest neighbor classification from multiple feature subsets. Intelligent Data Analysis, 3(3): 191-209, 1999.

[16] Kristin P. Bennett and Ayhan Demiriz and Richard Maclin, Exploiting unlabeled data in ensemble methods, Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 289–296, ACM Press, New York, NY, USA, 2002.

[17] Brazdil P., Gama J., Henery R., Characterizing the Applicability ofClassification Algorithms using Meta Level Learning, in Machine Learning:ECML-94, F.Bergadano e L. de Raedt (eds.), LNAI No. 784: pp. 83-102,Springer-Verlag, 1994.

[18] Breiman L., Bagging predictors, Machine Learning, 24(2):123-140, 1996.

[19] Breiman L., Arcing classifiers, Annals of Statistics, vol. 26,no. 3, pp. 801-849, 1998.

[20] Breiman, L., Pasting small votes for classification in large databases and on-line.Machine Learning, 36, 85-103.

[21] Breiman, L., Random forests. Machine Learn-ing, 45, 5-32, 2001.

[22] C.E. Brodley, Recursive automatic bias selection for classifier construction, Machine Learning 20 (1995) 63-94.

[23] R. Bryll, Gutierrez-Osuna R., Quek F., Attribute bagging: improving accuracy of classifier ensembles by using random feature subsets, Pattern Recognition Volume 36 (2003): 1291-1302

[24] Brown G., Wyatt J. L., Negative Correlation Learning and the Ambiguity Family of Ensemble Methods. Multiple Classifier Systems 2003: 266–275

[25] Brown G., Wyatt J., Harris R., Yao X., Diversity creation methods: a survey and categorisation, Information Fusion, 6(1):5–20.

[26] Bruzzone L., Cossu R., Vernazza G., Detection of land-cover transitions by combining multidate classifiers, Pattern Recognition Letters, 25(13): 1491–1500, 2004.

[27] Buntine, W., "Graphical Models for Discovering Knowledge", in U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, Advances in Knowledge Discovery and Data Mining, pp 59-82. AAAI/MIT Press, 1996.

[28] Buttrey, S.E., Karo, C., 2002. Using k-nearest-neighbor classification in the leaves of a tree. Comput. Statist. Data Anal. 40, 27-37.

[29] Caruana R., Niculescu-Mizil A. , Crew G. , Ksikes A., Ensemble selection from libraries of models, Twenty-first international conference on Machine learning, July 04-08, 2004, Banff, Alberta, Canada.

[30] Chan P.K. and Stolfo, S.J., A Comparative Evaluation of Voting and Meta-learning on Partitioned Data, Proc. 12th Intl. Conf. On Machine Learning ICML-95, 1995.

[31] Chan P.K. and Stolfo S.J, On the Accuracy of Meta-learning for Scalable Data Mining, J. Intelligent Information Systems, 8:5-28, 1997.

[32] Chawla N. V., Hall L. O., Bowyer K. W., Kegelmeyer W. P., Learning Ensembles from Bites: A Scalable and Accurate Approach, The Journal of Machine Learning Research archive, 5:421–451, 2004.

[33] Christensen S. W. , Sinclair I., Reed P. A. S., Designing committees of models through deliberate weighting of data points, The Journal of Machine Learning Research, 4(1):39–66, 2004.

[34] Christmann A., Steinwart I., Hubert M.,Robust learning from bites for data mining, Computational Statistics and Data Analysis 52 (2007) 347-361

[35] Clark, P. and Boswell, R., "Rule induction with CN2: Some recent improvements." In Proceedings of the European Working Session on Learning, pp. 151-163, Pitman, 1991.

[36] S. Cohen, L. Rokach, O. Maimon, Decision Tree input space Decomposition with Grouped Gain-Ratio, Information Sciences 177(17): 3592–3612, Elsevier, 2007.

[37] Croux C., Joossens K., Lemmens A., Trimmed bagging, Computational Statistics and Data Analysis 52 (2007) 362-368

[38] Cunningham P., and Carney J., Diversity Versus Quality in Classification Ensembles Based on Feature Selection, In: R. L. de Mntaras and E. Plaza (eds.), Proc. ECML 2000, 11th European Conf. On Machine Learning,Barcelona, Spain, LNCS 1810, Springer, 2000, pp. 109-116.

[39] B.V. Dasarathy and B.V. Sheela, "Composite classifier system design: Concepts and methodology," Proceedings of the IEEE, vol. 67, no. 5, pp. 708-713, 1979.

[40] Demsar J, Zupan B, Leban G (2004) Orange: From ExperimentalMachine Learning to Interactive Data Mining, White Paper(www.ailab.si/orange), Faculty of Computer and InformationScience, University of Ljubljana.

[41] Denison D.G.T., Adams N.M., Holmes C.C., Hand D.J., Bayesian partition modelling, Computational Statistics and Data Analysis 38 (2002) 475-485

[42] Derbeko P. , El-Yaniv R. and Meir R., Variance optimized bagging, European Conference on Machine Learning, 2002.

[43] Džeroski S., Ženko B., Is Combining Classifiers with Stacking Better than Selecting the Best One?, Machine Learning, 54(3): 255–273, 2004.

[44] Dietterich, T. G., and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. Journal of Artificial Intelligence Research, 2:263-286, 1995.

[45] Dietterich, T. G., An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting and Randomization. 40(2):139-157, 2000a.

[46] Dietterich T., Ensemble methods in machine learning. In J. Kittler and F. Roll, editors, First International Workshop on Multiple Classifier Systems, Lecture Notes in Computer Science, pages 1-15. Springer-Verlag, 2000b.

[47] Dimitrakakis C., Bengio S., Online adaptive policies for ensemble classifiers,Neurocomputing 64:211-221, 2005.

[48] Domingos, P., Using Partitioning to Speed Up Specific-to-General Rule Induction. In Proceedings of the AAAI-96 Workshop on Integrating Multiple Learned Models, pp. 29-34, AAAI Press, 1996.

[49] Drucker H., Effect of pruning and early stopping on performance of a boosting ensemble, Computational Statistics and Data Analysis 38 (2002) 393-406

[50] Duin, R. P. W., The combining classifier: to train or not to train? In Proc. 16th International Conference on Pattern Recognition, ICPR02, Canada, 2002, pp. 765-770.

[51] Elovici, Y., Shapira, B. and Kantor, P., Using the Information Structure Model to Compare Profile-Based Information Filtering Systems. Information Retrieval Journal 6(1), 2002, 75-97.

[52] Elovici, Y., Shapira, B., and Kantor, P., A Decision Theoretic Approach to Combining Information Filters: Analytical and Empirical Evaluation. Journal of the American Society for Information Science and Technology (JASIST), 57(3), 2006, 306-320.

[53] Frank E., Hall M.,Holmes G., Kirkby R., Pfahringer B., WEKA - A Machine LearningWorkbench for Data Mining, in O. Maimon, L. Rokach, editors TheData Mining and Knowledge Discovery Handbook, Springer, pp.1305-1314, 2005.

[54] Yoav Freund and Llew Mason. The Alternating Decision Tree Algorithm. Proceedings of the 16th International Conference on Machine Learning, pages 124-133 (1999)

[55] Freund Y. and Schapire R. E., Experiments with a new boosting algorithm. In Machine Learning: Proceedings of the Thirteenth International Conference, pages 325-332, 1996.

[56] Friedman, J. H., "Multivariate Adaptive Regression Splines", The Annual Of Statistics, 19, 1-141, 1991.

[57] Friedman, J., T. Hastie and R. Tibshirani (2000) Additive Logistic Regression: a Statistical View of Boosting, Annals of Statistics, 28(2):337-407.

[58] Friedman, J.H., 2002. Stochastic gradient boosting. Comput. Statist. Data Anal. 38 (4), 367-378.

[59] Gama J., A Linear-Bayes Classifier. In C. Monard, editor, Advances on Artificial Intelligence – SBIA2000. LNAI 1952, pp 269-279, Springer Verlag, 2000

[60] Gams, M., New Measurements Highlight the Importance of Redundant Knowledge. In European Working Session on Learning, Montpeiller, France, Pitman, 1989.

[61] Gey, S., Poggi, J.-M., 2006. Boosting and instability for regression trees Comput. Statist. Data Anal. 50, 533-550.

[62] Gunter S., Bunke H. , Feature Selection Algorithms for the generation of multiple classifier systems, Pattern Recognition Letters, 25(11):1323–1336, 2004.

[63] Hansen J., Combining Predictors. Meta Machine Learning Methods and Bias, Variance & Ambiguity Decompositions. PhD dissertation. Aurhus University. 2000.

[64] L.K. Hansen and P. Salamon, "Neural network ensembles," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 12, no. 10, pp. 993-1001, 1990.

[65] Ho T. K., The Random Subspace Method for Constructing Decision Forests, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 20, No. 8, 1998, pp. 832-844.

[66] T. K. Ho, Data complexity analysis for classifier combination, in: Proc. Int. Workshop on Multiple Classifier Systems (LNCS 2096), Springer, Cambridge, UK, 2001, pp. 53-67.

[67] Ho T. K., Multiple Classifier Combination: Lessons and Next Steps, in Kandel and Bunke, (eds.), Hybrid Methods in Pattern Recognition, World Scientific, 2002, 171–198.

[68] Ho T. K., Nearest Neighbors in Random Subspaces, Proc. of the Second International Workshop on Statistical Techniques in Pattern Recognition, Sydney, Australia, August 11-13, 1998, 640–648.

[69] Ho T. K. , Hull J.J., Srihari S.N.,Decision Combination in Multiple Classifier Systems, PAMI 1994, 16(1):66–75.

[70] Hothorn T., Lausen B., Bundling classifiers by bagging trees, Computational Statistics and Data Analysis 49 (2005) 1068-1078

[71] Hu, X., Using Rough Sets Theory and Database Operations to Construct a Good Ensemble of Classifiers for Data Mining Applications. ICDM01. pp. 233-240, 2001.

[72] Hu Q., Yu D., Xie Z., Li X., EROS: Ensemble rough subspaces,Pattern Recognition 40 (2007) 3728 - 3739.

[73] Huang Y. S. and Suen C. Y. , A method of combining multiple experts for the recognition of unconstrained handwritten numerals, IEEE Trans. Patt. Anal. Mach. Intell. 17 (1995) 90-94.

[74] Q. H. Hu, D. R. Yu, M. Y. Wang, Constructing Rough Decision Forests, D. Slezak et al. (Eds.): RSFDGrC 2005, LNAI 3642, Springer, 2005, pp. 147-156

[75] Mierswa, Ingo and Wurst, Michael and Klinkenberg, Ralf andScholz, Martin and Euler, Timm: YALE: Rapid Prototyping forComplex Data Mining Tasks, in Proceedings of the 12th ACM SIGKDDInternational Conference on Knowledge Discovery and Data Mining(KDD-06), 2006.

[76] M. M. Islam, X. Yao, K. Murase, A constructive algorithm for training cooperative neuralnetwork ensembles, IEEE Transactions on Neural Networks 14 (4)(2003) 820-834.

[77] Jacobs, R., Jordan, M., Nowlan, S., Hinton, G. 1991. Adaptivemixtures of local experts Neural Computation, 3, 79-87.

[78] Johansen T. A. and Foss B. A., A narmax model representation for adaptive control based on local model -Modeling, Identification and Control, 13(1):25-39, 1992.

[79] Jordan, M. I., and Jacobs, R. A., Hierarchical mixtures of experts and the EM algorithm. Neural Computation, 6, 181-214, 1994.

[80] Kamel M. S. and Wanas N. M., Data dependence in combining classifiers. In T. Windeattand F. Roli, editors, Proc. 4th Int. Workshop on Multiple Classifier Systems (MCS 2003),Vol. 2709 of Lecture Notes in Computer Science LNCS, Guildford, UK, 2003, Springer-Verlag, pp. 1-14.

[81] Kang H., Lee S., Combination Of Multiple Classifiers By Minimizing The Upper Bound Of Bayes Error Rate For Unconstrained Handwritten Numeral Recognition, International Journal of Pattern Recognition and Artificial Intelligence, 19(3):395 - 413, 2005.

[82] Kim,Y., Koo, J.-Y., 2005. Inverse boosting for monotone regression functions. Comput. Statist. Data Anal. 49, 757-770.

[83] Kohavi R., Scaling up the accuracy of naive-bayes classifiers: adecision-tree hybrid. In Proceedings of the Second InternationalConference on Knowledge Discovery and Data Mining, pages 114–119,1996.

[84] Kolen, J. F., and Pollack, J. B., Back propagation is sesitive to initial conditions. In Advances in Neural Information Processing Systems, Vol. 3, pp. 860–867 San Francisco, CA. Morgan Kaufmann, 1991.

[85] Krogh, A., and Vedelsby, J., Neural network ensembles, cross validation and active learning. In Advances in Neural Information Processing Systems 7, pp. 231–238 1995.

[86] Kuncheva L., Combining Pattern Classifiers, Wiley Press 2005.

[87] Kuncheva L.I. Diversity in multiple classifier systems (Editorial), Information Fusion, 6 (1), 2005, 3-4.

[88] Kuncheva, L., & Whitaker, C., Measures of diversity in classifier ensembles and their relationship with ensemble accuracy. Machine Learning 51:181–207, 2003.

[89] Kusiak, A., Decomposition in Data Mining: An Industrial Case Study, IEEE Transactions on Electronics Packaging Manufacturing, Vol. 23, No. 4, pp. 345-353, 2000.

[90] Lam L., Classifier combinations: implementations and theoretical issues. In J. Kittlerand F. Roli, editors, Multiple Classifier Systems, Vol. 1857 of Lecture Notes in ComputerScience, Cagliari, Italy, 2000, Springer, pp. 78-86.

[91] Langdon W. B., Barrett S. J., Buxton B. F., Combining decision trees and neural networks for drug discovery, in: Genetic Programming, Proceedings of the 5th European Conference, EuroGP 2002, Kinsale, Ireland, 2002, pp. 60–70.

[92] Leigh W., Purvis R., Ragusa J. M., Forecasting the NYSE composite index with technical analysis, pattern recognizer, neural networks, and genetic algorithm: a case study in romantic decision support, Decision Support Systems 32(4): 361–377, 2002.

[93] Jing Li, Nigel Allinson, Dacheng Tao, and Xuelong Li, Multitraining Support Vector Machine for Image Retrieval, IEEE Transactions on Image Processing, vol. 15, no. 11, pp. 3597-3601, November 2006.

[94] Liao Y., and Moody J., Constructing Heterogeneous Committees via Input Feature Grouping, in Advances in Neural Information Processing Systems, Vol.12, S.A. Solla, T.K. Leen and K.-R. Muller (eds.),MIT Press, 2000.

[95] Lin H., Kao Y., Yang F., Wang P., Content-Based Image Retrieval Trained By Adaboost For Mobile Application, International Journal of Pattern Recognition and Artificial Intelligence, 20(4):525-541, 2006.

[96] Lin L., Wang X., Yeung D., Combining Multiple Classifiers Based On A Statistical Method For Handwritten Chinese Character Recognition, International Journal of Pattern Recognition and Artificial Intelligence, 19(8):1027 - 1040, 2005.

[97] Liu H., Mandvikar A., Mody J., An Empirical Study of Building Compact Ensembles. WAIM 2004: pp. 622-627.

[98] Liu C., Classifier combination based on confidence transformation, Pattern Recognition 38 (2005) 11 - 28

[99] Lu B.L., Ito M., Task Decomposition and Module Combination Based on Class Relations: A Modular Neural Network for Pattern Classification, IEEE
Trans. on Neural Networks, 10(5):1244-1256, 1999.

[100] Maimon, O. and Rokach, L., Improving supervised learning by feature decomposition, Proceedings of the Second International Symposium on Foundations of Information and Knowledge Systems, Lecture Notes in Computer Science, Springer-Verlag, pp. 178–196, 2002.

[101] Maimon, O. and Rokach, L., Decomposition Methodology for Knowledge Discovery and Data Mining: Theory and Applications, Series in Machine Perception and Artificial Intelligence - Vol. 61, World Scientific Publishing, ISBN:981-256-079-3, 2005.

[102] Maimon, O. and Rokach, L., Data Mining by Attribute Decomposition with semiconductors manufacturing case study, Data Mining for Design and Manufacturing: Methods and Applications, pp. 311–336, 2001, Kluwer Academic Publishers.

[103] Mangiameli P., West D., Rampal R., Model selection for medical diagnosis decision support systems, Decision Support Systems, 36(3): 247–259, 2004.

[104] Margineantu D. and Dietterich T., Pruning adaptive boosting. In Proc. Fourteenth Intl. Conf. Machine Learning, pages 211–218, 1997.

[105] Prem Melville, Raymond J. Mooney: Constructing Diverse Classifier Ensembles using Artificial Training Examples. IJCAI 2003: 505-512

[106] Menahem, E., Shabtai, A., Rokach, L., Elovici, Y., Improving malware detection by applying multi-inducer ensemble. Computational Statistics and Data Analysis, 53(4):1483–1494, 2009.

[107] Menahem, E., Rokach, L., Elovici, Y., Troika - An Improved Stacking Schema for Classification Tasks, Information Sciences (to appear).

[108] Merkwirth C., Mauser H., Schulz-Gasch T., Roche O., Stahl M., Lengauer T., Ensemble methods for classification in cheminformatics, Journal of Chemical Information and Modeling, 44(6):1971–1978, 2004.

[109] Merler S., Caprile B., Furlanello C., Parallelizing AdaBoost by weights dynamics, Computational Statistics and Data Analysis 51 (2007) 2487-2498

[110] Merz, C. J., Using Correspondence Analysis to Combine Classifier, Machine Learning, 36(1-2):33-58, 1999.

[111] Michalski R. S., and Tecuci G.. Machine Learning, A Multistrategy Approach,Vol. J. Morgan Kaufmann, 1994.

[112] Mitchell, T., Machine Learning, McGraw-Hill, 1997.

[113] Moskovitch R, Elovici Y, Rokach L, Detection of unknown computer worms based on behavioral classification of the host, Computational Statistics and Data Analysis, 52(9):4544–4566, 2008.

[114] Nowlan S. J., and Hinton G. E. Evaluation of adaptive mixtures of competing experts. In Advances in Neural Information Processing Systems, R. P. Lippmann, J. E. Moody, and D. S. Touretzky, Eds., vol. 3, pp. 774-780, Morgan Kaufmann Publishers Inc., 1991.

[115] Opitz, D., Feature Selection for Ensembles, In: Proc. 16th National Conf. on Artificial Intelligence, AAAI,1999, pp. 379-384.

[116] Opitz, D. and Maclin, R., Popular Ensemble Methods: An Empirical Study, Journal of Artificial Research, 11: 169-198, 1999.

[117] Opitz D. and Shavlik J., Generating accurate and diverse members of a neuralnetwork ensemble. In David S. Touretzky, Michael C. Mozer, and Michael E. Hasselmo, editors, Advances in Neural Information Processing Systems, volume 8, pages 535–541. The MIT Press, 1996.

[118] Parmanto, B., Munro, P. W., and Doyle, H. R., Improving committee diagnosis with resampling techinques. In Touretzky, D. S., Mozer, M. C., and Hesselmo, M. E. (Eds). Advances in Neural Information Processing Systems, Vol. 8, pp. 882-888 Cambridge, MA. MIT Press, 1996.

[119] D. Partridge, W. B. Yates (1996), Engineering multiversion neural-net systems, Neural Computation, 8(4):869-893.

[120] Phama T., Smeuldersb A., Quadratic boosting, Pattern Recognition 41(2008): 331 - 341.

[121] Polikar R., "Ensemble Based Systems in Decision Making," IEEECircuits and Systems Magazine, vol.6, no. 3, pp. 21-45, 2006

[122] Prodromidis, A. L., Stolfo, S. J. and Chan, P. K., Effective and efficient pruning of metaclassifiers in a distributed Data Mining system. Technical report CUCS-017-99, Columbia Univ., 1999.

[123] Provost, F.J. and Kolluri, V., A Survey of Methods for Scaling Up Inductive Learning Algorithms, Proc. 3rd International Conference on Knowledge Discovery and Data Mining, 1997.

[124] Quinlan, J. R., Bagging, Boosting, and C4.5. In Proceedings of the Thirteenth National Conference on Artificial Intelligence, pages 725-730, 1996.

[125] Ricco Rakotomalala, "TANAGRA: a free software for research andacademic purposes", in Proceedings of EGC'2005, RNTI-E-3, vol. 2,pp.697-702, 2005

[126] R Development Core Team (2004), R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-00-3, http://cran.r-project.org/, 2004

[127] Ramamurti, V., and Ghosh, J., Structurally Adaptive Modular Networks for Non-Stationary Environments, IEEE Transactions on Neural Networks, 10 (1):152-160, 1999.

[128] Ridgeway G., Looking for lumps: boosting and bagging for density estimation,Computational Statistics and Data Analysis 38 (2002) 379-392

[129] Rokach L., Genetic algorithm-based feature set partitioning for classification problems,Pattern Recognition, 41(5):1676–1700, 2008.

[130] Rokach L., Mining manufacturing data using genetic algorithm-based feature set decomposition, Int. J. Intelligent Systems Technologies and Applications, 4(1):57-78, 2008.

[131] Rokach, L. and Maimon, O., Data mining for improving the quality of manufacturing: a feature set decomposition approach, Journal of Intelligent Manufacturing, 17(3):285–299, 2006, Springer.

[132] Rokach L., Decomposition Methodology for Classification Tasks - A Meta Decomposer Framework, Pattern Analysis and Applications, 9(2006):257-271.

[133] Rokach, L. and Maimon, O., Clustering methods, Data Mining and Knowledge Discovery Handbook, pp. 321–352, 2005, Springer.

[134] Rokach, L., Collective-agreement-based pruning of ensembles. Computational Statistics and Data Analysis, 53(4):1015–1026, 2009.

[135] Rokach, L. and Mainon, O., Theory and applications of attribute decomposition, Proceedings of IEEE International Conference on Data Mining (ICDM 01), IEEE Computer Society Press, 2001. pp 473–480

[136] Rokach L., R. Arbel, O. Maimon, "Selective Voting - Getting More For Less in Sensor Fusion", International Journal of PatternRecognition and Artificial Intelligence, 20(3):329-350, 2006.

[137] Rokach L., Chizi B. and Maimon O., A Methodology For Improving The Performance Of Non-Ranker Feature Selection Filters,International Journal of Pattern Recognition and Artificial Intelligence, 21(5): 809 - 830, 2007.

[138] Rokach L. and Maimon O., Feature Set Decomposition for Decision Trees, Journal of Intelligent Data Analysis, Volume 9, Number 2, 2005b, pp 131-158.

[139] Rokach L., Maimon O., Arad O., "Improving Supervised Learning by Sample Decomposition", International Journal of Computational Intelligence and Applications, 5(1):37-54, 2005.

[140] Rokach L., Maimon O. and Lavi I., Space Decomposition In Data Mining: A Clustering Approach, Proceedings of the 14th International Symposium On Methodologies For Intelligent Systems, Maebashi, Japan, Lecture Notes in Computer Science, Springer-Verlag, 2003, pp. 24–31.

[141] Rokach, L. and Averbuch, M. and Maimon, O., Information Retrieval System for Medical Narrative Reports, The 6th International Conference On Flexible Query Answering Systems, Lyon, France, Lecture Notes in Artificial intelligence 3055, pp. 217–228 Springer-Verlag, 2004

[142] Rokach L., Romano R., Maimon O., Negation Recognition in Medical Narrative Reports, Information Retrieval, 11(6): 499-538, 2008

[143] Rudin C., Daubechies I., and Schapire R. E., The Dynamics of Adaboost: Cyclic behavior and convergence of margins, Journal of Machine Learning Research Vol. 5, 1557-1595, 2004.

[144] Rosen B. E., Ensemble Learning Using Decorrelated Neural Networks. Connect. Sci. 8(3): 373-384 (1996)

[145] Schaffer, C., Selecting a classification method by cross-validation. Machine Learning 13(1):135-143, 1993.

[146] R.E. Schapire, "The strength of weak learnability," Machine Learning, vol. 5, no. 2, pp. 197-227, 1990.

[147] Schclar A., Rokach L.: Random Projection Ensemble Classifiers. ICEIS 2009: 309–316.

[148] Schclar A., Rokach L., A. Meisels, Ensemble Methods for Improving the Performance of Neighborhood-based Collaborative Filtering, Proc. ACM RecSys 2009 (to appear).

[149] Seewald, A.K. and Fürnkranz, J., Grading classifiers, Austrian research institute for Artificial intelligence, 2001.

[150] Seewald A.K.: How to Make Stacking Better and FasterWhile Also Taking Care of an Unknown Weakness, in Sammut C.,Hoffmann A. (eds.), Proceedings of the Nineteenth InternationalConference on Machine Learning (ICML 2002), Morgan Kaufmann Publishers, pp.554-561, 2002.

[151] Sexton J., Laake P., LogitBoost with errors-in-variables, Computational Statistics and Data Analysis 52 (2008) 2549-2559

[152] Sharkey, A., On combining artificial neural nets, Connection Science, Vol. 8, pp.299-313, 1996.

[153] Sharkey A., Sharkey N., Combining diverse neural networks, The Knowledge Engineering Review 12(3): 231–247, 1997.

[154] Sharkey, A., Multi-Net systems, In Sharkey A. (Ed.) Combining Artificial Neural Networks: Ensemble and Modular Multi-Net Systems. pp. 1-30, Springer
-Verlag, 1999.

[155] A. J. C. Sharkey, Types of multinet system, in: Proc. Int. Workshop on Multiple Classifier Systems (LNCS 2364), Springer, Calgiari, Italy, 2002, pp. 108-117.

[156] Shilen, S., Multiple binary tree classifiers. Pattern Recognition 23(7): 757-763, 1990.

[157] Skurichina M. and Duin R.P.W., Bagging, boosting and the random subspace method for linear classifiers. Pattern Analysis and Applications, 5(2):121–135, 2002

[158] Sohn S. Y., Choi, H., Ensemble based on Data Envelopment Analysis, ECML Meta Learning workshop, Sep. 4, 2001.

[159] Sohna S.Y., Shinb H.W., Experimental study for thecomparison of classifier combination methods, Pattern Recognition 40 (2007) 33 - 40.

[160] Sivalingam D., Pandian N., Ben-Arie J., Minimal Classification Method With Error-Correcting Codes For Multiclass Recognition, International Journal of Pattern Recognition and Artificial Intelligence 19(5): 663 - 680, 2005.

[161] Sun Y., Todorovic S., Li L. Reducing The Overfitting Of Adaboost By Controlling Its Data Distribution Skewness, International Journal of Pattern Recognition and Artificial Intelligence, 20(7):1093-1116, 2006.

[162] Tan A. C., Gilbert D., Deville Y., Multi-class Protein Fold Classification using a New Ensemble Machine Learning Approach. Genome Informatics, 14:206–217, 2003.

[163] Dacheng Tao, Xiaoou Tang, Xuelong Li, and Xindong Wu, Asymmetric Bagging and Random Subspace for Support Vector Machines-based Relevance Feedback in Image Retrieval, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 28, no.7, pp. 1088-1099, July 2006

[164] Dacheng Tao, Xuelong Li, and Stephen J. Maybank, Negative Samples Analysis in Relevance Feedback, IEEE Transactions on Knowledge and Data Engineering, vol. 19, no. 4, pp. 568-580, April 2007.

[165] Dacheng Tao and Xiaoou Tang, SVM-based Relevance Feedback Using Random Subspace Method, IEEE International Conference on Multimedia and Expo, pp. 647-652, 2004

[166] Towell, G. Shavlik, J., Knowledge-based artificial neural networks,Artificial Intelligence, 70: 119-165, 1994.

[167] Tsao, C.A., Chang, Y.I., 2007. A stochastic approximation view of boosting. Comput. Stat. Data Anal. 52 (1), 325-344.

[168] Tsymbal A., and Puuronen S., Ensemble Feature Selection with the Simple Bayesian Classification in Medical Diagnostics, In: Proc. 15thIEEE Symp. on Computer-Based Medical Systems CBMS2002, Maribor, Slovenia,IEEE CS Press, 2002, pp. 225-230.

[169] Tsymbal A., Pechenizkiy M., Cunningham P., Diversity in search strategies for ensemble feature selection. Information Fusion 6(1): 83-98, 2005.

[170] Tukey J.W., Exploratory data analysis, Addison-Wesley, Reading, Mass, 1977.

[171] Tumer, K. and Ghosh J., Error Correlation and Error Reduction in Ensemble Classifiers, Connection Science, Special issue on combining artificial neural networks: ensemble approaches, 8 (3-4): 385-404, 1996.

[172] Tumer, K., and Ghosh J., Robust Order Statistics based Ensembles for Distributed Data Mining. In Kargupta, H. and Chan P., eds, Advances in Distributed and Parallel Knowledge Discovery , pp. 185-210, AAAI/MIT Press, 2000.

[173] K. Tumer, C. N. Oza, Input decimated ensembles. Pattern Analysis and Application 6 (2003) 65-77.

[174] Tutz, G., Binder, H., 2006. Boosting ridge regression. Computational Statistics and Data Analysis. Corrected Proof, Available online 22 December 2006, in press.

[175] Valentini G. and Masulli F., Ensembles of learning machines. In R. Tagliaferri andM. Marinaro, editors, Neural Nets, WIRN, Vol. 2486 of Lecture Notes in ComputerScience, Springer, 2002, pp. 3-19.

[176] Vilalta R., Giraud–Carrier C., Brazdil P., "Meta-Learning", in O. Maimon and L. Rokach (Eds.), Handbook of Data Mining and Knowledge Discovery in Databases, pp. 731-748, Springer, 2005.

[177] Wanas Nayer M., Dara Rozita A. , Kamel Mohamed S., Adaptivefusion and co-operative training for classifier ensembles, PatternRecognition 39 (2006) 1781 - 1794

[178] Wang W., Jones P., Partridge D., Diversity between neural networks and decision trees for building multiple classifier systems, in: Proc. Int. Workshop on Multiple Classifier Systems (LNCS 1857), Springer, Calgiari, Italy, 2000, pp. 240–249.

[179] Webb G., MultiBoosting: A technique for combining boosting and wagging. Machine Learning, 40(2): 159-196, 2000.

[180] Webb G., and Zheng Z., Multistrategy Ensemble Learning: Reducing Error by Combining Ensemble Learning Techniques. IEEE Transactions on Knowledge and Data Engineering, 16 No. 8:980-991, 2004.

[181] Windeatt T. and Ardeshir G., An Empirical Comparison of Pruning Methods for Ensemble Classifiers, IDA2001, LNCS 2189, pp. 208-217, 2001.

[182] Woods K., Kegelmeyer W., Bowyer K., Combination of multiple classifiers using local accuracy estimates, IEEE Transactions on Pattern Analysis and Machine Intelligence 19:405–410, 1997.

[183] Wolpert, D.H., Stacked Generalization, Neural Networks, Vol. 5, pp. 241-259, Pergamon Press, 1992.

[184] Q. X. Wu , D. Bell and M. McGinnity, Multi-knowledge for decision making, Journal Knowledge and Information Systems, 7(2005): 246-266

[185] Xu L., Krzyzak A., Suen C.Y., Methods of combining multiple classifiers and their application to handwriting recognition, IEEE Trans. SMC 22, 418-435, 1992

[186] W. Yates, D. Partridge, Use of methodological diversity to improve neural network generalization,Neural Computing and Applications 4 (2) (1996) 114-128.

[187] Zenobi, G., and Cunningham, P. Using diversity in preparing ensembles of classifiers based on different feature subsets to minimize generalization error. In Proceedings of the European Conference on Machine Learning, 2001.

[188] Zhang, C.X., Zhang, J.S., 2008. A local boosting algorithm for solving classification problems. Comput. Stat. Data Anal. 52 (4), 1928-1941.

[189] Zhang, C.X., Zhang, J.S., Zhang G. Y., Using Boosting to prune Double-Bagging ensembles. Computational Statistics and Data Analysis, (2008), doi:10.1016/j.csda.2008.10.040

[190] Zhou, Z. H., and Tang, W., Selective Ensemble of Decision Trees, in Guoyin Wang, Qing Liu, Yiyu Yao, Andrzej Skowron (Eds.): Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing, 9^{th} International Conference, RSFDGrC, Chongqing, China, Proceedings. Lecture Notes in Computer Science 2639, pp.476-483, 2003.

[191] Zhou, Z. H., Wu J., Tang W., Ensembling neural networks: many could be better than all. Artificial Intelligence 137: 239-263, 2002.

[192] Zupan, B., Bohanec, M., Demsar J., and Bratko, I., Feature transformation by function decomposition, IEEE intelligent systems & their applications, 13: 38-43, 1998.

[193] Zhou Z., Jiang Y., NeC4.5: Neural Ensemble Based C4.5, IEEE Transactions on Knowledge and Data Engineering, vol. 16, no. 6, pp. 770-773, Jun., 2004.

[194] Zhoua J., Pengb H., Suenc C., Data-driven decomposition formulti-class classification, Pattern Recognition 41: 67 - 76, 2008.