Ensemble Approaches for Regression: A Survey

JOÃO MENDES-MOREIRA, LIAAD-INESC TEC, FEUP, Universidade do Porto
CARLOS SOARES, INESC TEC, FEP, Universidade do Porto
ALíPIO MÁRIO JORGE, LIAAD-INESC TEC, FCUP, Universidade do Porto
JORGE FREIRE DE SOUSA, INESC TEC, FEUP, Universidade do Porto

The goal of ensemble regression is to combine several models in order to improve the prediction accuracy in learning problems with a numerical target variable. The process of ensemble learning can be divided into three phases: the generation phase, the pruning phase, and the integration phase. We discuss different approaches to each of these phases that are able to deal with the regression problem, categorizing them in terms of their relevant characteristics and linking them to contributions from different fields. Furthermore, this work makes it possible to identify interesting areas for future research.

Categories and Subject Descriptors: I.2.6 [**Artificial Intelligence**]: Learning—*Induction*; *Knowledge acquisition*; *Parameter learning*

General Terms: Performance, Standardization

Additional Key Words and Phrases: Ensemble learning, multiple models, regression, supervised learning, neural networks, decision trees, support vector machines, k-nearest neighbors

ACM Reference Format:
Mendes-Moreira, J., Soares, C., Jorge, A. M., and de Sousa, J. F. 2012. Ensemble approaches for regression: A survey. ACM Comput. Surv. 45, 1, Article 10 (November 2012), 40 pages.
DOI = 10.1145/2379776.2379786 http://doi.acm.org/10.1145/2379776.2379786

1. INTRODUCTION

Ensemble learning typically refers to methods that generate several models that are combined to make a prediction, either in classification or regression problems. This approach has been the object of a significant amount of research in recent years and good results have been reported (e.g., Liu et al. [2000], Breiman [2001a], and Rodríguez et al. [2006]). The advantage of ensembles with respect to single models has been reported in terms of increased robustness and accuracy [García-Pedrajas et al. 2005].

Most work on ensemble learning focuses on classification problems. Unfortunately, successful classification techniques are often not directly applicable to regression. Therefore, although both are related, ensemble learning approaches for regression and classification have been developed somehow independently. As a consequence, existing surveys on ensemble methods for classification [Kuncheva 2004; Ranawana and Palade 2006; Polikar 2009; Rokach 2009b, 2009c, 2010] are not suitable for providing an overview of existing approaches for regression.

This work was partly funded by the Programa de Financiamento Plurianual de Unidades de I&D, by project Knowledge Discovery from Ubiquitous Data Streams (PTDC/EIA-EIA/098355/2008) and by project Rank! (PTDC/EIA/81178/2006).
Authors' addresses: J. Mendes-Moreira (corresponding author), C. Soares, A. M. Jorge, and J. F. de Sousa, Universidade do Porto, Portugal; email: jmoreira@fe.up.pt.

DOI 10.1145/2379776.2379786 http://doi.acm.org/10.1145/2379776.2379786

This article surveys existing approaches to ensemble learning for regression. The amount of work that has been published on ensemble regression is substantial and it is impossible to discuss all of it in one single work. The approach followed here was to identify the main trends and describe the key papers that represent some of the most representative ones for each of them. The relevance of this article is strengthened by the fact that ensemble learning is an object of research in different communities including pattern recognition, machine learning, statistics, and neural networks. These communities have different conferences and journals and often use different terminology and notation. This makes it quite hard for a researcher to be aware of all contributions that are relevant to his or her own work. Therefore, in addition to attempting to provide a thorough account of the work in the area, we also organize those approaches independently of the research area they were originally proposed in. Furthermore, this format is adequate for both classification and regression problems. Hopefully, this format will make it possible to identify opportunities for further research and facilitate the classification of new approaches.

Although the information available is insufficient to identify definite rules concerning which methods to use for which regression problems, this article provides some guidelines that can help practitioners on method selection.

In the next section, we provide a general discussion of the ensemble learning process. This discussion will lay the foundations for the remaining sections of the article: ensemble generation (Section 3), ensemble pruning (Section 4) and ensemble integration (Section 5). Each of these three sections discusses aspects that are specific to the corresponding step. Section 6 complements this by discussing general issues. Finally, Section 7 concludes the article with a summary.

2. ENSEMBLE LEARNING FOR REGRESSION

This section presents the regression problem. Following this, a more accurate definition of ensemble learning and the associated terminology are outlined. Additionally, this section presents a general description of the process of ensemble learning and describes a taxonomy of different approaches, both of which define the structure of the rest of the article. The experimental setup for the evaluation of ensemble learning proposals is then discussed. This is followed by an analysis of the error decomposition of ensemble learning methods for regression. A complete example of an ensemble method for regression concludes this section.

2.1. Regression

In this work we assume a typical regression problem. With a potentially infinite input space X, the goal is to induce a function $\hat{f} : X \rightarrow \Re$ that approximates an unknown true function f. The quality of the approximation is given by the generalization error, which is typically defined as

$$mse(\hat{f}) = E[(\hat{f} - f)^2]. \tag{1}$$

The function $\hat{f}$ is obtained by running an induction algorithm (or learner) on data consisting of a finite set of n examples of the form $\{(\mathbf{x_1}, f(\mathbf{x_1})), \ldots, (\mathbf{x_n}, f(\mathbf{x_n}))\}$. The $\hat{f}$ function is called a model or predictor. Then, given that it is not possible to determine the true error of a model $\hat{f}$ according to Eq. (1), the error is estimated on a different set of data, consisting of n_{test} examples (see Section 2.3).

$$mse(\hat{f}) \approx \frac{1}{n_{test}} \sum_{i=1}^{n_{test}} [\hat{f}(\mathbf{x_i}) - f(\mathbf{x_i})]^2 \tag{2}$$

Many other generalization error functions exist for numerical predictions [Witten and Frank 2011] that can also be used for ensemble regression. However, most of the work on ensemble regression uses mse, so this issue will not be discussed here.

2.2. Definition of Ensemble Learning

First of all, it is important to clearly define what ensemble learning is and to define a taxonomy of methods. Some of the existing definitions are partial in the sense that they only focus on the classification problem or on part of the ensemble learning process [Dietterich 1997]. For these reasons the following definition is proposed.

> Ensemble learning is a process that uses a set of models, each of them obtained by applying a learning process to a given problem. This set of models (ensemble) is integrated in some way to obtain the final prediction.

This definition has important characteristics. Firstly, contrary to the informal definition given at the beginning of the article, this definition not only covers ensembles in supervised learning (both classification and regression problems), but also in unsupervised learning, namely the ensemble clustering research, also known as consensual clustering [Strehl and Ghosh 2003; Monti et al. 2003].

Additionally, it clearly separates ensemble and divide-and-conquer approaches. This last family of approaches splits the input space into several subregions and trains each model separately in each one of the subregions. With this approach the initial problem is converted into several simpler subproblems.

Finally, it does not separate the combination and selection approaches which most definitions do. According to this definition, selection is a special case of combination where all of the weights are zero except for one of them (to be discussed in Section 5).

2.2.1. The Ensemble Learning Process.

The ensemble process can be divided into three steps [Roli et al. 2001] (Figure 1), that are usually referred to as the overproduce-and-choose approach. The first step is *ensemble generation*, which consists of generating a set of models (Section 3). A number of redundant models are often generated during the first step. In the *ensemble pruning* step, the ensemble is pruned by eliminating some of the models generated earlier (Section 4). Finally, in the *ensemble integration* step, a strategy to combine the base models is defined. This strategy is then used to obtain the prediction of the ensemble (represented as $\hat{f}_F$) for new cases, based on the predictions of the base models (Section 5).

Our characterization of the ensemble learning process is slightly more detailed than the one presented by Rooney et al. [2004]. For those authors, ensemble learning consists of the solution of two problems: (1) how to generate the ensemble of models (ensemble generation); and (2) how to integrate the predictions of the models from the ensemble in order to obtain the final ensemble prediction (ensemble integration). This last approach (without the pruning step), is called direct and can be seen as a particular case of the model presented in Figure 1, labeled overproduce-and-choose. In some cases ensemble pruning has been reported to reduce the size of the ensembles obtained without degrading the accuracy. Pruning has also been added to direct methods successfully, increasing the accuracy [Zhou et al. 2002; Martinez-Munoz et al. 2009].

Ensemble regression is characterized by the use of predictors to address the regression problem, as defined in Section 2.1.

2.2.2. Taxonomy and Terminology.

With regard to the categorization of the different approaches to ensemble learning, the taxonomy used will be the one presented by the same authors [Rooney et al. 2004]. They divide ensemble generation approaches into *homogeneous*, if all of the models were generated using the same induction algorithm and *heterogeneous* otherwise.

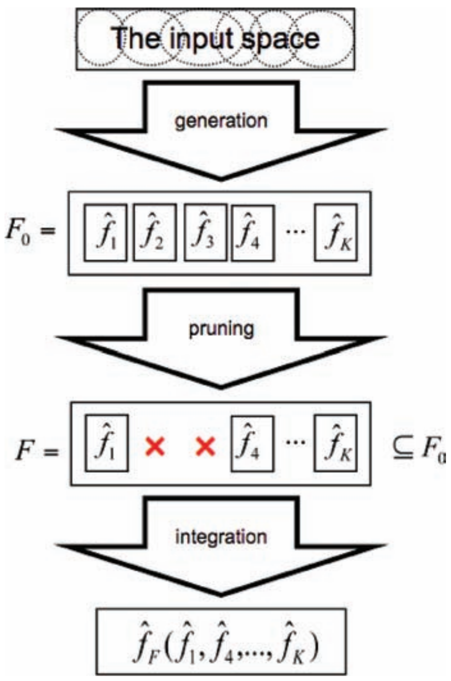

The input space
generation
$F_0 = \hat{f}_1 \; \hat{f}_2 \; \hat{f}_3 \; \hat{f}_4 \cdots \hat{f}_K$
pruning
$F = \hat{f}_1 \; \times \; \times \; \hat{f}_4 \cdots \hat{f}_K \subseteq F_0$
integration
$\hat{f}_F(\hat{f}_1, \hat{f}_4, \ldots, \hat{f}_K)$

Fig. 1. Ensemble learning model.

Table I. Synonyms

ensemble	committee, multiple models, multiple classifiers (regressors)
predictor	model, regressor (classifier), learner, hypothesis, expert
example	instance, case, data point, object
combination	fusion, competitive classifiers (regressors), ensemble approach, multiple topology
selection	cooperative classifiers (regressors), modular approach, hybrid topology

Ensemble integration methods are classified by some authors [Rooney et al. 2004; Kuncheva 2002] as *combination* (also called *fusion*) or as *selection*. The former approach combines the predictions of the models from the ensemble in order to obtain the final ensemble prediction. The latter approach selects the most promising model(s) from the ensemble and the prediction of the ensemble is only based on the selected model(s). Here the classification of constant versus nonconstant weighting functions given by Merz [1998] has been used instead. In the first case, the predictions of the base models are always combined in the same way. In the second case, the way the predictions are combined can be different for different input values.

As mentioned earlier, research on ensemble learning is carried out in different communities. Therefore, different terms are sometimes used for the same concept. Table I lists several groups of synonyms that have been adapted from a previous list by Kuncheva [2004]. The first column contains the most frequently used terms in this article.

2.3. Experimental Setup

The experimental setup used to test the ensemble learning methods differs significantly in different papers. Since the goal of this work is not to survey experimental setups, the most common setup will be described. It is based on splitting the data into three parts: (1) the training set $\mathcal{L}$, used to obtain the base predictors; (2) the validation set, used to assess the generalization error of the base predictors; and (3) the test set, used to assess the generalization error of the final ensemble method. If a pruning algorithm is used, it is tested together with the integration method on the test set. Hastie et al. [2001] suggest splitting the set; 50% for training, 25% for validation, and the remaining 25% to use as a test set. This strategy works for large datasets, but as stated by Hastie et al., it is too difficult to state a general rule on how much training data is necessary.

As a general principle, for shorter datasets, the possibility of using resampling methods should be considered, such as cross-validation [Stone 1974], for instance, in order to increase the number of examples in the training set.

For datasets that can be used to evaluate the research conducted on ensemble regression, it is important to use problems that are widely available. This makes it possible to reproduce previous results and perform a more reliable comparison of methods. Several repositories collect regression datasets. The most popular is the UCI machine learning repository. However, only 15 of its 200 datasets are regression problems [Frank and Asuncion 2010]. Another interesting repository that only archives regression datasets is the one maintained by Luís Torgo [Torgo 2011]. Torgo's repository also has a small but carefully selected list of links to other repositories of regression datasets. Another possible source of regression datasets is the homepage for the Delve project [Delve 2002]. It has both real and synthetically generated datasets. All of these repositories identify the regression datasets clearly. In other repositories, this distinction is not so clear. Examples of such repositories are: the datasets archive from StatLib [Vlachos 2005], the function approximation repository from Bilkent University, Turkey [Guvenir and Uysal 2000], the repository from the Machine Learning Group at UCD School of Computer Science and Informatics, Dublin, Ireland [MLG 2011], the Datamob datasets collection that has 255 datasets [Flannagan and Sperber 2008], or the Data Wrangling that announces 400 datasets, some of which are not free of charge [Skomoroch 2008]. It should be noted that many of the datasets are repeated in different repositories. The list of repositories presented is not intended to be complete.

2.4. Understanding the Generalization Error of Ensembles

In order to accomplish the task of ensemble generation, the characteristics that the ensemble should have must be known. Empirically, one can argue that a successful ensemble is one with accurate predictors and that makes errors in different parts of the input space [Perrone and Cooper 1993]. Therefore, understanding the generalization error of ensembles is essential to knowing which characteristics the predictors should have in order to reduce the overall generalization error. The decomposition of the generalization error in regression is straightforward. In the following, a few alternative ways of decomposing the *mse* (Eq. (2)) are presented that closely follow Brown's [2004] description. Despite the fact that most of these decompositions were originally proposed for neural network ensembles, they are not dependent on the induction algorithm used. The functions are represented, when appropriate, without the input variables, just for the sake of simplicity. For example, instead of $f(\mathbf{x})$, f is used.

Geman et al. present the bias/variance decomposition for a single neural network [Geman et al. 1992].

$$E\{[\hat{f} - E(f)]^2\} = [E(\hat{f}) - E(f)]^2 + fE\{[\hat{f} - E(\hat{f})]^2\} \tag{3}$$

The first term on the right-hand side is called the bias and represents the distance between the expected value of the estimator $\hat{f}$ and the unknown population average. The second term, the variance component, measures how the predictions vary with respect to the average prediction. This can be rewritten as

$$mse(f) = bias(f)^2 + var(f). \tag{4}$$

Krogh and Vedelsby describe the ambiguity decomposition, for an ensemble of K neural networks [Krogh and Vedelsby 1995]. Assuming that the prediction of the ensemble is the weighted sum of the individual predictions, $\hat{f}_{\mathcal{F}}(\mathbf{x}) = \sum_{i=1}^{K}[\alpha_i \times \hat{f}_i(\mathbf{x})]$ (see Section 5.1) where $\sum_{i=1}^{K}(\alpha_i) = 1$ and $\alpha_i \geq 0, i = 1, \ldots, K$, they show that the error for a single example is

$$(\hat{f}_{\mathcal{F}} - f)^2 = \sum_{i=1}^{K}[\alpha_i \times (\hat{f}_i - f)^2] - \sum_{i=1}^{K}[\alpha_i \times (\hat{f}_i - \hat{f}_{\mathcal{F}})^2]. \tag{5}$$

This expression shows explicitly that the ensemble generalization error is less than or equal to the generalization error of a randomly selected single predictor. This is true because the ambiguity component (the second term on the right) is always nonnegative. Another important result of this decomposition is that it is possible to reduce the ensemble generalization error by increasing the ambiguity without increasing the bias. The ambiguity term measures the disagreement among the base predictors on a given input **x** (omitted in the formulae just for the sake of simplicity, as previously mentioned). Two full examples that prove the ambiguity decomposition [Krogh and Vedelsby 1995] are presented in Brown [2004].

More recently, Ueda and Nakano presented the bias/variance/covariance decomposition of the generalization error of ensemble estimators [Ueda and Nakano 1996]. In this decomposition it is assumed that $\hat{f}_{\mathcal{F}}(\mathbf{x}) = \frac{1}{K} \times \sum_{i=1}^{K}[\hat{f}_i(\mathbf{x})]$. We have

$$E[(\hat{f}_{\mathcal{F}} - f)^2] = \overline{bias}^2 + \frac{1}{K} \times \overline{var} + \left(1 - \frac{1}{K}\right) \times \overline{covar}, \tag{6}$$

where

$$\overline{bias} = \frac{1}{K} \times \sum_{i=1}^{K}[E_i(f_i) - f], \tag{7}$$

$$\overline{var} = \frac{1}{K} \times \sum_{i=1}^{K}\{E_i\{[\hat{f}_i - E_i(\hat{f}_i)]^2\}\}, \tag{8}$$

$$\overline{covar} = \frac{1}{K \times (K-1)} \times \sum_{i=1}^{K} \sum_{j=1, j \neq i}^{K} E_{i,j}\{[\hat{f}_i - E_i(\hat{f}_i)][\hat{f}_j - E_j(\hat{f}_j)]\}. \tag{9}$$

The indexes i, j of the expectation mean that the expression is true for particular training sets, respectively, $\mathcal{L}_i$ and $\mathcal{L}_j$.

Brown et al. provide a good discussion of the relationship between ambiguity and covariance [Brown et al. 2005b]. An important result obtained from the study of this relationship is the confirmation that it is not possible to maximize the ensemble ambiguity without affecting the ensemble bias component as well. This means that it is not possible to maximize the ambiguity component and minimize the bias component simultaneously.

The discussion of the present section is usually referred to in the context of ensemble diversity, that is, the study of the degree of disagreement between the base predictors. Many of the preceding statements are related to the well-known statistical problem of point estimation [Lehmann 1998]. This discussion is also related to the multicollinearity problem that will be discussed in Section 5. A more detailed discussion on these issues can be found in Brown et al. [2005b].

```
Require: $\mathcal{L} = \{(\mathbf{x_i}, y_i), i = 1, 2, \ldots, n\}$, the dataset
Require: $\mathcal{B}$, the base learning algorithm
Require: $K_0$, the intended number of models
Require: $PK$, percentage of models from the pool that will be used for prediction
 1: for $i := 1$ to $K_0$ do
 2:    $\mathcal{L}_i = Bootstrap(\mathcal{L})$
 3:    $\hat{f}_i = \mathcal{B}(\mathcal{L}_i)$
 4: end for
 5: for $i := 1$ to $K_0$ do
 6:    for $j := 1$ to $K_0$ do
 7:       $C_{i,j} := \frac{1}{n}\sum_{p=1}^{n}[(\hat{f}_i(\mathbf{x_p}) - f(\mathbf{x_p}))(\hat{f}_j(\mathbf{x_p}) - f(\mathbf{x_p}))]$
 8:    end for
 9: end for
10: $s$ := empty vector
11: for $i := 1$ to $K$ do
12:    minimum := $+\infty$
13:    for $j \in \{1$ to $K_0\} \setminus \{s_1, \ldots, s_{i-1}\}$ do
14:       value := $i^{-2}(\sum_{p=1}^{i-1}\sum_{q=1}^{i-1} C_{s_p,s_q} + 2\sum_{p=1}^{i-1} C_{s_p,j} + C_{j,j})$
15:       if (value < minimum) then
16:          $s_i$ := $j$
17:          minimum := value
18:       end if
19:    end for
20: end for
21: $\mathcal{F} = \{\hat{f}_i | i = s_1, s_2, ..., s_K\}$
22: return $\mathcal{F}$
```

Fig. 2. The pseudo-code of bagging [Breiman 1996a] with ordered pruning [Hernández-Lobato et al. 2006].

2.5. A Detailed Example of an Ensemble Method for Regression

This section sketches the algorithm of a simple ensemble method for regression. It provides a framework for the rest of the article that will be of interest for those readers who are less familiar with ensemble learning, particularly in the context of regression. It is based on one of the earliest and most popular ensemble methods: bagging [Breiman 1996a]. The pruning phase, proposed by Hernández-Lobato et al. [2006], is added in order to reduce the ensemble size without meaningfully reducing the accuracy of the ensemble predictions. This is an example of an ensemble method for regression with all of the three phases described previously: generation, pruning, and integration.

The generation phase of bagging has three parameters: the dataset $\mathcal{L}$, the base learning algorithm $\mathcal{B}$, and the number of models that will be generated K_0 ($K_0 = 100$ is often used). An important assumption of bagging is that the base learning algorithm is sensitive to variations in the training set. For this reason, bagging implementations are usually based on decision trees or artificial neural networks, which are two unstable algorithms. Bagging takes advantage of the instability by training the models using random samples (with replacement) of the dataset, named bootstrap samples. The size of the samples is equal to the size of the original dataset. The instability of the base learning algorithm guarantees that the ensemble generated has the properties of accuracy and diversity that were previously discussed. The generation step is described in steps 1 to 4 in Figure 2.

The pruning step aims to reduce the ensemble size without significantly reducing the accuracy of the ensemble. The algorithm proposed by Hernández-Lobato et al. [2006]

```
Require: $\mathcal{F}$, the ensemble
Require: x, the example to make a prediction about
1: $K := Size(\mathcal{F})$
2: return $\frac{1}{K}\sum_{i=1}^{K}\hat{f}_i(\mathbf{x})$
```

Fig. 3. Ensemble integration using simple average.

has been used. It has a single parameter, K, that defines the number of models that will be selected. Hernandez-Lobato et al. suggest that $K \approx 0.2K_0$. In summary, the method selects the top K models with higher contribution to the accuracy of the ensemble. Pruning goes from step 5 to step 21 of Figure 2. The first part of this algorithm (steps 5 to 9) computes the covariance matrix for the K_0 predictors. Then, the algorithm uses a sequential forward search method to obtain a subset of K predictors (steps 10 to 20). At each iteration, it selects the regressor from the pool that, when incorporated, reduces the training error of the ensemble the most. The criterion used (step 14) for the selection of the predictors favors the selection of accurate predictors with diversity in the errors, as discussed in Section 2.4.

With regard to the last step, the integration function we use is the same that is used in the original bagging algorithm [Breiman 1996a]: the simple average. The integration function is not represented in Figure 2 because, while generation and pruning is a onetime process, integration is performed for each prediction. The pseudocode of the integration step is presented in Figure 3.

This example shows one specific approach for each step of the ensemble learning process in detail. The remainder of this article surveys other approaches for each step.

3. ENSEMBLE GENERATION

As previously mentioned, the first step in ensemble learning is the ensemble generation (Figure 1). The goal of ensemble generation is to obtain a set of models.

$$\mathcal{F}_0 = \{\hat{f}_i, i = 1, \ldots, K_0\} \tag{10}$$

If the models are generated using the same induction algorithm, the ensemble is called homogeneous, otherwise it is called heterogeneous.

Homogeneous ensemble generation is the area of ensemble learning best covered in the literature. See, for example, the state-of-the-art surveys from Dietterich [1997], or Brown et al. [2005a]. This section mainly follows the former [Dietterich 1997]. In homogeneous ensembles, the models are generated using the same algorithm. Thus, as detailed in the following sections, accurate and diverse predictors can be achieved by manipulating the data (Section 3.1) or through the model generation process (Section 3.2).

Heterogeneous ensembles are obtained when more than one learning algorithm is used. This approach is expected to obtain models with more diversity [Webb and Zheng 2004] due to the different nature of the base learners. Of course, in order to guarantee a low generalization error for the ensemble, the base learners must be as accurate as possible, as previously discussed in Section 2.4. One problem is the lack of control over the diversity of the base learners during the generation phase. In homogeneous ensembles, diversity can be systematically controlled during their generation, as will be discussed in the following sections. Conversely, when using several algorithms, it may not be so easy to control the differences between the generated models. This difficulty can be solved by using the overproduce-and-choose approach. By generating a large number of models (the "overproduce" step), the probability of obtaining an accurate and diverse subset of predictors increases. The task of selecting that subset is then

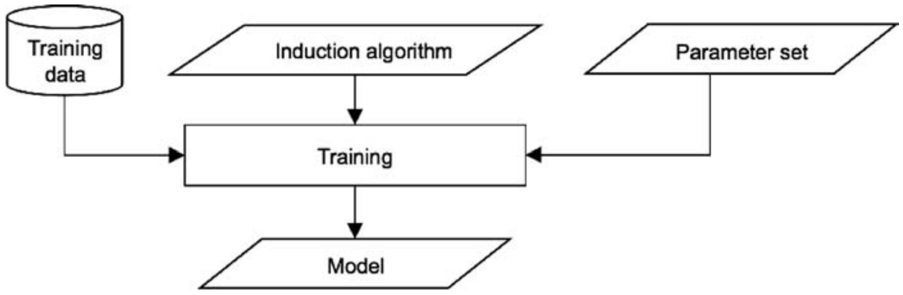


Fig. 4. The generation process.

left to the pruning phase (the “choose” step) [Caruana et al. 2004]. Another approach that is commonly followed combines the two approaches by using different induction algorithms mixed with the use of different parameter sets [Merz 1996; Rooney et al. 2004] (Section 3.2.1). Some authors claim that heterogeneous ensembles perform better than homogeneous ensembles [Wichard et al. 2003]. Note that heterogeneous ensembles can use homogeneous ensemble models as base learners. Another approach that is commonly followed combines the two approaches by using different induction algorithms mixed with the use of different parameter sets [Merz 1996; Rooney et al. 2004] (Section 3.2.1). Some authors claim that heterogeneous ensembles obtain better performance than homogeneous ensembles [Wichard et al. 2003]. Note that heterogeneous ensembles can use homogeneous ensemble models as base learners.

This section presents methods for ensemble generation. These methods are classified according to whether they manipulate the data or the modeling process in order to generate different, and typically diverse, models. The modeling process can be manipulated through the induction algorithm, the set of input parameters, or the generated model. This is described in Figure 4.

3.1. Data Manipulation

Data can be manipulated in three different ways: subsampling from the training set, manipulating the input features, and manipulating the output variable.

3.1.1. Subsampling from the Training Set. This approach generates models using different subsamples from the training set and assuming that the algorithm is unstable, that is, small changes in the training set imply important changes in the result. Decision trees, neural networks, rule learning algorithms, and MARS are well-known unstable algorithms [Breiman 1996b; Dietterich 1997]. However, some of the methods based on subsampling (e.g., bagging and boosting) have been successfully applied to algorithms that are usually regarded as stable, such as Support Vector Machines (SVM) [Kim et al. 2003].

One of the most popular of such methods is bagging [Breiman 1996a] (already described in Section 2.5). It uses randomly generated training sets to obtain an ensemble of predictors. If the original training set $\mathcal{L}$ has M examples, bagging (bootstrap aggregating) generates a model by uniformly sampling M examples with replacement (some examples appear several times while others do not appear at all). Both Breiman [1996a] and Domingos [1997] give insights on why bagging works.

Some attempts have been made to reduce the computational cost of bagging. This is the case of subagging, which obtains each base model using a random subset of the examples [Buhlmann 2010]. Using decision trees as base learners, Buja and Stuetzle [2006] report similar results between bagging and subagging when using subsamples with half the size of the original training set.

Parmanto et al. describe the *cross-validated committees* technique for neural networks ensemble generation using υ-fold cross-validation [Parmanto et al. 1996]. The main idea is to create an ensemble with the models obtained from the υ training sets of a cross-validation process.

Another important family of subsampling-based ensemble methods is boosting, which is discussed in the remainder of this subsection. Based on Schapire [1990], Freund and Schapire present the AdaBoost (ADAptive BOOSTing) algorithm, the most popular *boosting algorithm* [Freund and Schapire 1996]. The main idea is that it is possible to convert a weak learning algorithm into one that arbitrarily achieves high accuracy. A weak learning algorithm is one that performs slightly better than random prediction. This conversion is performed by combining the estimations of several predictors. Like in bagging [Breiman 1996a], the examples are randomly selected with replacement but, in AdaBoost, each example has a different probability of being selected. Initially, this probability is equal for all of the examples, but in the following iterations, examples with more inaccurate predictions have a higher probability of being selected. In each new iteration there are more "difficult examples" in the training set. Boosting was originally developed for classification and is not directly applicable to regression because the function that updates the weights at each iteration assumes that the generalization error is $\epsilon \in]0.5, 1]$.

The first attempt to adapt the AdaBoost algorithm to regression was AdaBoost.R [Freund and Schapire 1997]. This method assumes that $y \in [0, 1]$. It transforms the regression dataset into a binary classification dataset as follows: (1) the range of the target value is split into a given number S of equal-sized intervals, with lower limits $l_1 = 1/(S+1), l_2 = 2/(S+1), \ldots, l_S = S/(S+1)$; (2) each instance i is replaced by S of its copies, where the target value of copy j is replaced with two new variables. The first is a new independent variable with value l_j and the second is the new target variable of the binary classification problem, defined as 0 if $y_i < l_j$ or 1, otherwise. Therefore, each new instance j associated with the original instance i represents the question "is $y_i < l_j$?" The new dataset has $N * S$ instances. This approach has two main problems [Duffy and Helmbold 2002]: (1) the increase of the computational cost due to the increase of the dataset size; and (2) the instability of the loss function across iterations and even between instances.

Avnimelech and Intrator propose a very simple adaptation of AdaBoost by changing the error function used internally. At each boosting iteration, the regression errors of the models obtained in the previous iteration are discretized into two classes. In other words, when the regression error of an example is higher than a given threshold, the classification error is set to 1, otherwise, it is set to 0 [Avnimelech and Intrator 1999]. The remainder of the algorithm they propose is identical to AdaBoost. An improvement to this work, called AdaBoost.RT, is presented in Shrestha and Solomatine [2006]. At each iteration the error is calculated using only a subset of the examples. This subset contains the examples with an error higher than a given threshold. In Schclar et al. [2009], an alternative example selection rule is proposed. The subset of examples used to compute the error contains the ones with an error that is higher than the mean of the errors, plus a given deviation factor, multiplied by the standard deviation of the errors. The advantage of this modification is that the threshold is calculated according to the statistics of the errors. Since this is completed at each iteration, the threshold changes at each iteration.

The AdaBoost.R2 [Drucker 1997] algorithm follows a different approach to ensuring that the error function behaves as expected in the original AdaBoost algorithm. It uses different functions to transform the error values, guaranteeing that the average (transfomed) error for all of the instances in the training set is within the interval $[0, 1]$. The process finishes when the (transformed) average error is lower than 0.5.

This approach was tested using different base learners (DART [Friedman 1996], PPR [Friedman and Stuetzle 1981], and MARS [Friedman 1991]) [Borra and Ciaccio 2002]. The best base learners on four datasets were DART and PPR.

Friedman [2001] developed an approach which is quite different from the previous ones but embodies the boosting spirit of focusing the learning process on the examples that were incorrectly predicted by previous models in a nice way. Instead of assigning higher weights to those examples, the target variable of the dataset at each iteration is replaced by the residuals of the predictions made by the model of the previous iteration. The ensemble obtained by using this type of approach is also known as a forward stage-wise additive model. This method was later improved by using bootstrap samples at each iteration [Friedman 2002]. This improvement avoids overfitting by evaluating the error at each iteration in the out-of-bag set. This last version is known as stochastic gradient boosting and is commercialized under the name Multiple Additive Regression Trees (MART). It must be noted that it is arguable that this approach can be classified as an ensemble learning method according to our definition (Section 2.2), because only the first predictor addresses the original learning problem directly (i.e, the remaining models predict the residuals of the previous model). However, we consider that they address the original problem, even if indirectly.

In Zemel and Pitassi [2001], a regression-specific version of AdaBoost is proposed that combines the objective function presented by Drucker [1997] with a gradient-based approach [Friedman 2001; Duffy and Helmbold 2002] rather than the original additive approach.

One of the difficulties of developing theoretically founded ensemble approaches to regression is the infinite size of the model space. In Ratsch et al. [2002] a boosting-like approach is proposed where the weight updating function is based on the dual of a linear programming problem. Thus, it is independent of the number of models. Besides the strong theoretical foundation, the method obtains good results on a few time-series. However, some open issues remain, including a comparison to other ensemble methods.

3.1.2. Manipulating the Input Features. In this approach, different training sets are obtained by changing the representation of the examples. A new training set $\mathcal{L}'$ is generated by replacing the original representation $\{(\mathbf{x_i}, f(\mathbf{x_i}))\}$ with a new one $\{(\mathbf{x'_i}, f(\mathbf{x_i}))\}$. There are two types of approaches. The first one is feature selection, that is, $\mathbf{x'_i} \subset \mathbf{x_i}$. In the second approach, the representation is obtained by applying some transformation to the original attributes, that is, $\mathbf{x'_i} = g(\mathbf{x_i})$.

A simple feature selection approach is the *random subspace* method [Ho 1998]. The models in the ensemble are independently constructed using randomly selected feature subsets. Decision trees were originally used as base learners and the ensemble was called *decision forests* [Ho 1998]. The final prediction is the combination of the predictions of all of the trees in the forest. In Schclar and Rokach [2009] feature extraction is used instead of feature selection. The original features are projected into a new space by randomly combining them. The new set of features is smaller than the original one. Tests using k-nearest neighbors as base learners show a slightly better performance than random subspaces.

Alternatively, iterative search methods can be used to select the different feature subsets. Opitz uses a genetic algorithm approach that continuously generates new subsets, starting with a randomly selected subset [Opitz 1999]. The author tests the approach on classification problems using neural networks as a learning algorithm. The criterion used to select the feature subsets is the minimization of the individual error and the maximization of ambiguity (Section 2.4). He reports better results using this approach than using the popular bagging and AdaBoost methods. A similar approach is proposed in Zenobi and Cunningham [2001]. Both are wrapper-based approaches but, while Opitz [1999] searches for the best solution using a genetic algorithm, in Zenobi and Cunningham [2001] a hill-climbing strategy is used instead.

A feature selection approach can also be used to generate ensembles for algorithms that are stable with respect to the training set, but unstable with respect to the set of features, namely the nearest-neighbors induction algorithm. In Domeniconi and Yan [2004], feature subset selection is combined with adaptive sampling [Thompson and Seber 1996] to reduce the risk of discarding useful information. When compared to random feature selection, this approach reduces diversity between base predictors but increases their accuracy.

A simple transformation approach is *input smearing* [Frank and Pfahringer 2006]. It aims to increase the diversity of the ensemble by adding Gaussian noise to the inputs. The goal is to improve the results of bagging. Each input value x is changed into a smeared value x' using

$$x'_i = x_i + p * N(0, \hat{\sigma}_X), \tag{11}$$

where p is an input parameter of the input smearing algorithm and $\hat{\sigma}_X$ is the sample standard deviation of X, using the training set data. In this case, the examples are changed, but the training set keeps the same number of examples. Although only the numeric input variables are smeared in this work, similar strategies could be used for the nominal ones. Results presented in this work compare favorably to bagging. A similar approach called BEN (*Bootstrap Ensemble with Noise*) was previously presented by Raviv and Intrator [1996].

An alternative transformation approach uses feature discretization [Cai and Wu 2008]. In this work, numerical features are replaced with discretized versions. By repeating the process multiple times and varying the discretization method and its parameters, different datasets are generated. This promotes the diversity of the models generated.

Rodriguez et al. [2006] present a method that combines selection and transformation, called *rotation forests*. The original set of features is divided into k disjoint subsets to increase the chance of obtaining increased diversity. Then, for each subset, a Principal Component Analysis (PCA) approach is used to project the examples into a set of new features consisting of linear combinations of the original ones. Using decision trees as base learners, this strategy promotes diversity (decision trees are sensitive to the rotation of the axis) and accuracy (PCA generates features representing most of the information contained in the data). The rotation forest method has also been used for regression [Zhang et al. 2008]. In experiments on two real and three artificial datasets, it obtained worse results than AdaBoost.R2, similar results as bagging, and better results than random forest.

3.1.3. Manipulating the Output Variable. Manipulation of the output target values can also be used to generate different training sets. However, not much research follows this approach and most of it focuses on classification.

An exception is the work of Breiman, called *output smearing* [Breiman 2000]. The basic idea is to add Gaussian noise to the target variable of the training set, in the same way as is done for input features in the input smearing method (Section 3.1.2). Although it was originally proposed using CART trees as base models, it can also be used with other base algorithms. The comparison between output smearing and bagging shows a consistent reduction of the generalization error, even if not outstanding.

An alternative approach consists of the following steps. First a model is generated using the original data. This model is applied to the training data and a new training set is obtained by replacing the target values with the errors of this model. Using the new dataset, it generates a model that estimates the error of the predictions of the first model. Then, it creates a first ensemble that combines the prediction of the first model with the correction predicted by the second one. Finally, it starts an iterative process that: (1) generates models that predict the error of the current ensemble and (2) updates the ensemble with the new model. The training set used to generate the new model in each iteration is obtained by replacing the output targets with the errors of the current ensemble. This approach was proposed by Breiman, using bagging as the base algorithm and it was called *iterated bagging* [Breiman 2001b]. Iterated bagging reduces generalization error when compared to bagging. This is mainly due to bias reduction during the iterative process.

3.2. Modeling Process Manipulation

As an alternative to manipulating the training set, it is possible to change the model generation process. This can be performed by using different parameter sets, by manipulating the induction algorithm, or by manipulating the resulting model.

3.2.1. Manipulating the Parameter Sets. Every induction algorithm is sensitive to the values of the input parameters. The degree of sensitivity of an algorithm is different for different input parameters. These parameters can be manipulated in order to obtain ensembles with diverse and accurate predictors.

Neural network ensemble approaches quite often use different initial weights to obtain different models. This is done because the resulting models may vary significantly with different initial weights [Kolen and Pollack 1990]. Several authors, such as Rosen, for example, use randomly generated seeds (initial weights) to obtain different models [Rosen 1996], while other authors combine this strategy with the use of different number of layers and hidden units [Perrone and Cooper 1993; Hashem 1993].

The k-nearest-neighbors ensemble proposed by Yankov et al. [2006] only has two members. They differ on the number of nearest neighbors used. They are both suboptimal, one of them because the number of nearest neighbors is too small, and the other because it is too large. The purpose is to increase diversity (see the use of areas of expertise described in Section 5).

3.2.2. Manipulating the Induction Algorithm. Diversity can also be achieved by changing the way induction is performed. Therefore, the same learning algorithm may have different results on the same data. Two main categories of approaches for this can be identified as: sequential and parallel. In sequential approaches, the induction of a model is only influenced by the previous ones. In parallel approaches it is possible to have more extensive collaboration during the learning process: (1) each process takes into account the overall quality of the ensemble and (2) information about the models is exchanged between processes.

The most common sequential approach consists of changing the error function (e.g., Rosen [1996], Granitto et al. [2005], and Islam et al. [2003]). Rosen [1996] generates ensembles by sequentially training neural networks. The error function that is optimized during the training of a network includes a decorrelation penalty. Using this approach, the training of each network tries to minimize the covariance component of the error of the ensemble, according to Ueda and Nakano [1996] (Section 2.4), thus decreasing its generalization error and increasing diversity. This was the first approach using the decomposition of the generalization error made by Ueda and Nakano [1996] (Section 2.4) to guide the ensemble generation process. SECA (*Stepwise Ensemble Construction Algorithm*) in addition uses bagging to obtain the training set for each neural network [Granitto et al. 2005]. It stops when adding another neural network to the current ensemble increases the generalization error. The *Cooperative Neural Network Ensembles* (CNNE) method [Islam et al. 2003] begins with two neural networks and then iteratively adds new networks to try to minimize the ensemble error. As in Rosen's approach, the error function includes a term that represents the correlation between the models in the ensemble. Therefore, to stimulate diversity, all of the models already generated are trained again at each iteration of the process. This means that this is simultaneously sequential and parallel. Before adding new networks, this approach starts by adding hidden nodes to them. Therefore, it combines the manipulation of the algorithm with the manipulation of the model (Section 3.2.3). These methods were tested on both classification and regression datasets with promising results.

A different approach is proposed by Tsang et al. [2006]. They adapt the CVM (Core Vector Machines) algorithm [Tsang et al. 2005] to maximize the diversity of the models in the ensemble by guaranteeing that they are orthogonal. This is achieved by adding constraints to the quadratic programming problem that is solved using the CVM algorithm. This approach can be related to AdaBoost because higher weights are given to instances that have been incorrectly classified in previous iterations (Section 3.1.1).

Parallel approaches have, typically, three main characteristics: (1) simultaneous training of the models; (2) use of an adapted error function; and (3) search with evolutionary approaches.

The models are trained simultaneously but the learning processes are not independent. They interact to guarantee that the training of each model is trying to accomplish global objectives, that is, concerning the overall ensemble. Interaction is typically achieved by using an adapted error function. As in sequential approaches, this error function has a penalty term that guarantees the diversity of the ensemble. Evolutionary approaches are commonly used to obtain the right values for the penalty terms. In summary, the main difference between the parallel approaches and the sequential ones is that the ensemble generation is performed simultaneously taking into account (in each model of the ensemble) the behavior of the other models in previous iterations, weighted by a penalty term.

In the ADDEMUP (*Accurate anD Diverse Ensemble-Maker giving United Predictions*) method [Opitz and Shavlik 1996], the fitness metric for each network weighs the accuracy of the network and the diversity of this network within the ensemble, according to the bias/variance decomposition [Krogh and Vedelsby 1995]. Genetic operators of mutation and crossover are used to generate new models from previous ones. The new networks are trained emphasizing misclassified examples, as in AdaBoost (Section 3.1.1). The best networks are selected and the process is repeated until a stopping criterion is met. This approach can be used on other induction algorithms.

The method *Ensemble Learning via Negative Correlation* (ELNC) [Liu and Yao 1999] also learns the neural networks simultaneously. However, the error function uses a negative correlation term, according to Ueda and Nakano [1996], instead of the bias/variance decomposition from Krogh and Vedelsby [1995] used in Opitz and Shavlik [1996]. A combination of ELNC with an evolutionary programming framework, called *Evolutionary Ensembles with Negative Correlation Learning* (EENCL), was later proposed [Liu et al. 2000]. In this case, the only genetic operator used is mutation, which randomly changes the weights of an existing neural network. Furthermore, the ensemble size is obtained automatically.

A parallel approach is one in which each learning process does not take the quality of the others into account, but in which the exchange of information about the models is given by the *cooperative coevolution of artificial neural network ensembles* method [García-Pedrajas et al. 2005]. It also uses an evolutionary approach to generate ensembles of neural networks. It combines a mutation operator that affects the weights of the networks, as in EENCL, with another which affects their structure, as in ADDEMUP. As in EENCL, the generation and integration of models are also part of the same process. The diversity of the models in the ensemble is encouraged in two ways: (1) by using a coevolution approach, in which subpopulations of models evolve independently; and (2) by using a multiobjective evaluation fitness measure, combining network and ensemble fitness. Other groups of objectives (measures) besides the cooperation objectives are: performance objectives, regularization, diversity, and ensemble objectives. The authors conduct a study on the sensitivity of the algorithm to changes in the set of objectives. The results are interesting but they cannot be generalized to the regression problem, since the authors only studied the classification problem. This approach can be used for regression, but with a different set of objectives.

Finally we mention two other parallel techniques. In the first one, the learning algorithm generates the ensemble directly. Lin and Li formulate an *infinite ensemble* based on the SVM (Support Vector Machines) algorithm [Lin and Li 2005]. The main idea is to create a kernel that embodies all of the possible models in the hypothesis space. The SVM algorithm is then used to generate a linear combination of all of those models, which is, in fact, an ensemble of an infinite set of models. They propose the *stump kernel* that represents the space of decision stumps.

Breiman's *random forests* method [Breiman 2001a] uses an algorithm for the induction of decision trees that is also modified to incorporate some randomness: the split used at each node takes into account a randomly selected feature subset. The subset considered in one node is independent of the subset considered in the previous one. This strategy, based on the manipulation of the learning algorithm, is combined with subsampling, since the ensemble is generated using the bagging approach (Section 3.1). The strength of the method is the combined use of bootstrap sampling and random feature selection.

3.2.3. Manipulating the Model. When given a learning process that produces a single model f, it can potentially be transformed into an ensemble approach by producing a set of models f_i from the original model f. Jorge and Azevedo have proposed a postbagging approach for classification [Jorge and Azevedo 2005] that takes a set of Classification Association Rules (CAR's) produced by a single learning process and obtains n models by repeatedly sampling the set of rules. Predictions are obtained using a large committee of classifiers constructed as described before. Experimental results on 12 datasets show a consistent, although small, advantage over the singleton learning process. The same authors also propose an approach with some similarities to boosting [Azevedo and Jorge 2007]. Here, the rules in the original model f are iteratively reassessed, filtered, and reordered according to their performance on the training set. Once again, experimental results show a small but consistent improvement over using the original model, and also show a reduction in the bias component of the error. Both approaches replicate the original model without relearning and obtain very homogeneous ensembles with a kind of jittering effect around the original model [Azevedo and Jorge 2010]. Model manipulation has only been applied in the realm of classification association rules, a highly modular representation. Applying it to other kinds of models, such as decision trees or neural networks, does not seem trivial. However, it could be easily attempted with regression rules.

3.3. Discussion on Ensemble Generation

Two relevant issues arise from the previous discussion. The first is how can the user decide which method to use on a given problem. The second, which is more interesting from a researcher's point of view, is what are the promising lines for future work.

It is possible to distinguish the most interesting/promising methods from some of the most commonly used induction algorithms. For regression trees, bagging [Breiman 1996a], due to its consistency and simplicity, and random forest [Breiman 2001a], due to its accuracy, are the most appealing ensemble methods. For neural networks, the methods based on negative correlation (e.g., EENCL [Liu et al. 2000]) are particularly appealing, due to their theoretical foundations [Brown et al. 2005b] and successful empirical results. Islam et al. [2003] and Garcia-Pedrajas et al. [2005] also present interesting methods: the first one because it builds the base learners and the ensemble simultaneously, thus saving time when compared to generating all of the models and then combining them; the second one because it integrates operational research methods into the ensemble learning process.

Although ensembles are mostly used with unstable algorithms, it has been shown that process manipulation approaches (Section 3.2) obtain good results with stable algorithms [Ferrer et al. 2009].

With regard to the second issue, an important line of work is the adaptation of the methods described here to other algorithms, such as Support Vector Regression (SVR) and k-Nearest Neighbors (k-NN). k-NN is an unstable algorithm so it would be interesting to understand better exactly how it behaves with different ensemble generation techniques. SVR is stable so it does not make sense to build ensembles using the data manipulation approaches (Section 3.1). Furthermore, it has many parameters, including the kernel function, that significantly affect its results. Therefore, it would be interesting to analyze its performance with different modeling process manipulation approaches (Section 3.2). Although some attempts have been made along these lines, there is still much work to be done.

It can also be observed that there is less work on heterogeneous ensembles than in homogeneous ones. This is because it is more difficult to control the interaction between the different learning processes. Given that the quality of ensembles is linked to the accuracy and diversity of the models, combining models from different algorithms seems to be a promising approach in order to achieve diversity. The accuracy of the models should be ensured by its adequate choice.

Additionally, it must be noted that most research focuses on one specific approach to build the ensemble (e.g., subsampling from the training set or manipulating the induction algorithm). The successful results obtained by random forests indicate that mixing different generation processes is a promising approach [Meyer et al. 2003]. However, further investigation into the advantages of combining several approaches is necessary.

In general, the observations made here with regard to regression also apply to classification. One exception is when the induction algorithm is manipulated and in the boosting approach. Approaches that manipulate the induction algorithm take advantage of the error decomposition, which is naturally different in regression and in classification. Boosting algorithms for regression must be different due to the different nature of the generalization error for classification and for regression.

Tables II and III present a summary of some methods for ensemble generation. Table IV presents some real examples where some of these methods are used.

We conclude this section by summarizing pair-wise evaluations between different methods on ensemble generation for regression (Table V). Although such studies are useful, they should be read carefully. In fact, even though ensemble methods typically have a small number of parameters that meaningfully affects the ensemble's accuracy, these parameters should be previously tuned in order to obtain the best of each method. This is done in Meyer et al. [2003] but not in Zhang et al. [2008] and Yu et al. [2007], for instance. And the question is: is this the reason why bagging beats random forests in 5 out of 5 datasets in Zhang et al. [2008], but loses 12 out of 12 datasets in Meyer et al. [2003]? Additionally, it would be even more important to understand under which conditions do the methods perform best and worst. One approach on this kind of analysis is metalearning [Brazdil et al. 2009]. The results of such an analysis would not only support users in selecting the most suitable model for a given problem, but would also provide important insights for researchers in this area concerning, among others, which characteristics of each method should be improved.

Table II. Summary on Methods for Ensemble Generation

Abbrev.	Name	Ref.	Base learners
	Bagging	[Breiman 1996a]	DT & ANN
	Random subspace	[Ho 1998]	DT
	Ensemble feature selection	[Opitz 1999]	ANN
	Using diversity in preparing ensembles	[Zenobi and Cunningham 2001]	ANN
	Nearest Neighbor Ensemble	[Domeniconi and Yan 2004]	KNN
BEN	Bootstrap Ensemble with Noise, or Input smearing	[Raviv and Intrator 1996] [Frank and Pfahringer 2006]	DT
	Ensembles based on RSBRA	[Cai and Wu 2008]	SVM
	Rotation forests	[Rodríguez et al. 2006] [Zhang et al. 2008]	DT
	Output smearing	[Breiman 2000]	DT
	Iterated bagging	[Breiman 2001b]	DT
	AdaBoost.R2	[Drucker 1997]	DT
	AdaBoost.RT	[Shrestha and Solomatine 2006]	DT
MART	Multiple Additive Regression Trees	[Friedman 2002]	DT
SECA	Stepwise Ensemble Construction Algorithm	[Rosen 1996]	ANN
CNNE	Cooperative Neural Network Ensembles	[Islam et al. 2003]	ANN
CVM	Core Vector Machines	[Tsang et al. 2006]	SVM
ADDEMUP	Accurate anD Diverse Ensemble-Maker giving United Predictions	[Opitz and Shavlik 1996]	ANN
EENCL	Evolutionary Ensembles with Negative Correlation Learning	[Liu et al. 2000]	ANN
	Cooperative coevolution of neural network ensembles	[García-Pedrajas et al. 2005]	ANN
	Infinite ensemble	[Lin and Li 2005]	SVM
	Random forests	[Breiman 2001a]	DT
	Jittering ensembles	[Jorge and Azevedo 2005] [Azevedo and Jorge 2010]	AR

DT: Decision Trees; ANN: Artificial Neural Networks; KNN: K-Nearest neighbors; SVM: Support Vector Machines; AR: Association Rules.

4. ENSEMBLE PRUNING

Many of the previously discussed methods on ensemble generation generate diverse ensembles although they do not guarantee the use of the smallest ensemble capable of maximum accuracy. Some of those generation processes involve randomness, for instance, varying the training set, but they cannot foresee how diverse are the generated models.

Ensemble pruning consists of selecting a subset $\mathcal{F}$ of the models generated in the previous step, $\mathcal{F}_0$ (Eq. (10)). In this circumstance, $\mathcal{F}_0$ is also known as *pool of models*.

$$\mathcal{F} \subseteq \mathcal{F}_0 \tag{12}$$

The aim of ensemble pruning is to improve its predictive ability or reduce costs. It can also be used to avoid the problem of multicollinearity [Perrone and Cooper 1993; Hashem 1993] (to be discussed in Section 5). This is the "choose" step in the overproduce-and-choose approach. Even in the case of approaches that are designed to be direct (i.e., in which all the models are originally intended to be used, e.g., bagging

Table III. Summary on Methods for Ensemble Generation (continuation)

Name	Approach	Description
Bagging	SS	Constructs trees randomly selecting instance subsets using bootstrapping
AdaBoost.R2	SS	Adapts the AdaBoost classification algorithm by mapping the absolute error to [0, 1]
AdaBoost.RT	SS	Adapts the AdaBoost classification algorithm by discretizing the error in a binary one
Multiple additive regression trees	SS	Adapts the AdaBoost classification using a forward stage-wise additive model
Random subspace	MIF	Constructs trees randomly selecting features
Ensemble feature selection	MIF	Uses genetic algorithms for feature selection
Using diversity in preparing ensembles	MIF	Uses wrapper hill-climbing for feature subset selection
Nearest neighbor ensemble	MIF	Combines feature subset selection with adaptive sampling
Input smearing, or Bootstrap ensemble with noise	MIF	Constructs different trees adding Gaussian noise to the inputs
Ensembles based on RSBRA	MIF	Generates different models by transforming the input variables using random discretization
Rotation forests	MIF	Constructs trees randomly selecting feature subsets but retaining some features in all subsets using principal component analysis
Output smearing	MOV	Constructs different trees adding Gaussian noise to the output
Iterated bagging	MOV	Iteratively adds trees to the ensemble by training the trees using the error obtained with the current ensemble
Stepwise ensemble construction algorithm	MIA	Uses a decorrelation penalty in the error function during the sequential training of the neural networks
Cooperative neural network ensembles	MIA	It also uses a correlation term, but all neural networks are retrained whenever a new network is added to the ensemble
Core vector machines	MIA	Manipulates the quadratic problem of SVMs by adding constraints in order to guarantee orthogonality
Accurate and diverse ensemble-maker giving united predictions	MIA	Uses genetic algorithms for the generation of new models from previous ones emphasizing misclassified examples
Evolutionary ensembles with negative correlation learning	MIA	Uses a negative correlation term according to the bias/variance/covariance decomposition combined with evolutionary programming
Cooperative coevolution of neural network ensembles	MIA	Uses a multi-objective method in order to favor the cooperation among networks
Infinite ensemble	MIA	Uses a kernel for SVM that embodies all possible models in the hypothesis space
Random forests	MIA	Uses a random subset of the input features at each split in the tree construction together with bootstrapping for example selection
Jittering ensembles	MM	Uses sub-sampling over the rules of a single base model

SS: Sub-Sampling; MIF: Manipulating the Input Features; MOV: Manipulating the Output Variable; MIA: Manipulating the Induction Algorithm; MM: Manipulating the Model.

[Breiman 1996a]), it has been shown that the addition of a pruning step may not only reduce computational costs, but also increase prediction accuracy in some cases [Zhou et al. 2002; Hernández-Lobato et al. 2006]. Additionally, some pruning approaches (e.g., Bakker and Heskes [2003]) generate profiles for the different types of classifiers. These profiles summarize the knowledge in the ensembles, thus providing new insight on the data [Bakker and Heskes 2003].

Table IV. Summary on Methods for Ensemble Generation (applications)

Name	Applications	Refs
Bagging	e.g.: project management	[Aggarwal et al. 2010]
	real estate appraisals	[Lasota et al. 2009]
	image processing	[Meng et al. 2010]
Cooperative neural network ensembles	e.g.: Mechanics	[Zhang et al. 2009]
Evolutionary ensembles with negative correlation learning	e.g.: Telecommunications	[Yao et al. 2001]
Random forests	e.g.: biology	[Tian et al. 2010]
	software engineering	[Weyuker et al. 2010]
	transportation	[Mendes-Moreira et al. 2012]

Table V. Pair-wised Comparison between Ensemble Generation Methods for Regression

GM	BL	LWin/RWin	GM	BL	References
Bagging	DART	1/1	AdaBoost.R2	DART	[Borra and Ciaccio 2002]
Bagging	PPR	0/2	AdaBoost.R2	PPR	[Borra and Ciaccio 2002]
Bagging	CART	1/1	Iterated bagging	CART	[Breiman 2001b]
Bagging	CART	1/3	Subagging	CART	[Buja and Stuetzle 2006]
Bagging	CART	1/7	AdaBoost.R2	CART	[Drucker 1997; Zhang et al. 2008]
Bagging	ANN	0/5	SECA	ANN	[Granitto et al. 2005]
Bagging	ANN	5/43	Bagging	LR	[Yu et al. 2007]
Bagging	ANN	6/41	Random forest	C4.5	[Yu et al. 2007]
Bagging	LR	22/26	Random forest	C4.5	[Yu et al. 2007]
Bagging	CART	5/12	Random forest	CART	[Zhang et al. 2008; Meyer et al. 2003]
Bagging	CART	3/2	Rotation forest	CART	[Zhang et al. 2008]
Random forest	CART	0/5	AdaBoost.R2	CART	[Zhang et al. 2008]
Random forest	CART	0/5	Rotation forest	CART	[Zhang et al. 2008]
AdaBoost.R2	CART	5/0	Rotation forest	CART	[Zhang et al. 2008]
Bagging	CART	4/8	MART	CART	[Meyer et al. 2003]
Random forest	CART	10/2	MART	CART	[Meyer et al. 2003]

GM: Generation Method; BL: Base Learner; LWin: the method on the left is the winner; RWin: the method on the right is the winner; Draws were ignored.
DART: Recursive covering algorithm [Friedman 1996]; PPR: Projection Pursuit Regression; CART: Classification And Regression Trees; ANN: Artificial Neural Networks; LR: Logistic Regression; C4.5: C4.5 decision tree.

Even though there are surveys on ensemble pruning [Tsoumakas et al. 2008; Martinez-Munoz et al. 2009], they only address classification. However, some of the approaches used for classification cannot be applied to regression. The pruning algorithm described in Section 2.5 is an example of that. It uses the decomposition of the generalization error for regression, which is different from the one for classification. Moreover, parts of the ensemble pruning algorithms for regression are necessarily different from those for classification, namely the evaluation measures.

It is possible to make a comparison between pruning ensembles and feature selection [Molina et al. 2002], even though there are differences between the two tasks. Therefore, we draw some inspiration from feature selection to characterize existing methods on ensemble pruning.

Methods for ensemble pruning can be classified as partitioning-based or as search-based. Partitioning-based methods divide the pool of models into subgroups using a given partitioning criterion. For ensemble pruning only the use of clustering is known as a partitioning method. Then, for each subgroup one or more models are selected using a given selection criterion. Partitioning-based methods are discussed in Section 4.1.

Search-based methods search for a subset of the original pool of models by iteratively adding or removing models from the candidate subset according to a given evaluation measure and search algorithm (i.e., a strategy to add or remove models from the candidate subset). Search-based methods can be classified according to: (1) the object of evaluation; (2) the search algorithm; and (3) the evaluation measure. As far as (1) is concerned, single models or subsets of models can be evaluated. The search algorithm (2) depends on the object of the evaluation. The approaches that evaluate single models are known as ranking approaches. For ranking approaches the search is exhaustive, which means that all the base models are evaluated. On the other hand, the approaches that evaluate subset of models can use exhaustive, randomized, or sequential search algorithms, as will be discussed in Section 4.2.

The studies on the evaluation measures for ensemble pruning (3) are strongly related to the studies on the generalization error of the ensembles (Section 2.4). Evaluation measures are discussed in Section 4.3.

Additional classification criteria could be used in order to characterize search-based ensemble pruning methods. The stopping criterion defines when search should be stopped. It is *direct* if the number of models to be selected is given or *by evaluation* if it depends on the quality of the ensemble (single model or subset of models).

4.1. Partitioning-Based Approaches

This approach assumes that the pool contains many similar models and only a few of them are not redundant. The main idea of partitioning-based approaches is to divide the models into several subgroups using a partitioning criterion and to choose representative models (one or more) from each subgroup.

All the partitioning-based approaches take into consideration the generation of the subgroups using clustering algorithms. Lazarevic represents the models in the pool according to the prediction vectors made by them [Lazarevic 2001]. The k-means clustering algorithm is used on these vectors to obtain clusters of similar models. The number of clusters is an input parameter of this approach. In practice, this value must be tested by running the algorithm for different values or, as in Lazarevic's case, using an algorithm to obtain a good default value [Lazarevic 2001]. Coelho and Von Zuben use the ARIA - Adaptive Radius Immune Algorithm [Coelho and Von Zuben 2006] for clustering. This algorithm does not require the number of clusters to be prespecified [Bezerra et al. 2005].

As far as the selection of the models from the partitions is concerned, the goal is, as previously discussed, to generate the best possible ensemble, which typically means that the selected models must be accurate and diverse. In partitioning-based methods this is achieved, at least partially, while generating the subgroups of models. By choosing models from different groups, some diversity is guaranteed in the ensemble. Consequently, in practice, evaluation measures for the selection component of partitioning-based approaches are typically different (and simpler) than the ones used in the search-based approaches, which usually favor the accuracy of the ensemble. Evaluation measures are discussed in Section 4.3.

4.2. Search-Based Approaches for Model Subset Selection

Here, the same classification of algorithms that was used for feature subset selection are used [Aha and Bankert 1996; Molina et al. 2002]: exponential, randomized, and sequential.

4.2.1. Exponential Search Algorithms. Exponential algorithms search the complete input space. When selecting a subset of models from a pool $\mathcal{F}_l$ with K models, the search space has $2^K - 1$ nonempty subsets. The search for the optimal subset is an NP-complete problem [Tamon and Xiang 2000]. Perrone and Cooper suggest this approach for small values of K [Perrone and Cooper 1993]. However, according to Martínez-Muñoz and Suárez it is intractable for values of $K > 30$ [Martinez-Munoz and Suárez 2006].

An example of this approach is presented in Aksela [2003] where the evaluation measure is calculated for each nonempty candidate subsets. They use a pool of eight models.

4.2.2. Randomized Search Algorithms. Randomized algorithms perform a heuristic search in the input space using stochastic methods, such as evolutionary algorithms.

Ruta and Gabrys use three randomized algorithms to search for the best subset of models [Ruta and Gabrys 2001]: genetic algorithms, tabu search, and population-based incremental learning. The main result of the experiments on three classification datasets, using a pool of $K = 15$, was that the three algorithms obtained most of the best selectors comparatively to the exhaustive search. These results may have been conditioned by the small size of the pool.

4.2.3. Sequential Search Algorithms. Sequential algorithms iteratively change one solution by adding or removing models. Three types of sequential search algorithms are used according to how the successors of a solution are generated [Molina et al. 2002].

—Forward: The search begins with an empty ensemble. Models are added to the ensemble in each iteration. This is referred to as Forward Subset Selection (FSS).
—Backward: The search begins with all the models in the ensemble. Models from the ensemble are eliminated in each iteration. This is referred to as Backward Subset Selection (BSS).
—Combined: If the selection can have both forward and backward steps, it is called combined.

In the FSS and BSS algorithms, the stopping criterion assumes that the evaluation measure is monotonic [Coelho and Von Zuben 2006]. However, in practice, this cannot be guaranteed.

An interesting sequential forward search method for pruning ensembles is based on AdaBoost [Martinez-Munoz and Suárez 2007]. Instead of learning a new model in each iteration, this method selects the one from the initial pool of models that minimizes the error function. By changing the weights of the instances, as in the original AdaBoost approach, this method ensures that the selected models are diverse as well as accurate. Although this method was originally proposed for classification, it can be directly applied to regression using AdaBoost.R or AdaBoost.R2 (described in Section 3.1.1). Furthermore, the use of gradient descent boosting algorithms, if possible, is not direct due to their additive nature.

A rather different and interesting approach is based on frequent itemsets [Zhao et al. 2009]. Here, a pattern mining algorithm is adapted so as to iteratively build the final ensemble by adding accurate subsets of base models. It depends on two important classification-specific issues: a 0-1 loss function and integration with majority voting. One possibility of adapting this method for regression is to discretize the error into two classes as discussed previously (Section 3.1.1).

Combined methods address this difficulty by mixing forward and backward steps. The aim is to avoid local minima, that is, situations where the fast improvement obtained in the initial iterations leads to a solution which is not the best globally. Examples of combined search methods for ensemble pruning are Moreira et al. [2006] and Margineantu and Dietterich [1997].

Moreira et al. describe an algorithm that starts by randomly selecting a predefined number of K models [Moreira et al. 2006]. At each iteration, one forward step and one backward step are applied. The forward step is the same as in common FSS methods, that is, it selects the model from the pool which improves the accuracy of the ensemble the most. At the end of this step, the ensemble has $K + 1$ models. The second step selects the K models with higher ensemble accuracy. This means that, in practice, one of the $K + 1$ models is removed from the ensemble. The process stops when the same model is selected in both steps.

Margineantu and Dietterich present another compound search algorithm called reduce-error pruning with back fitting [Margineantu and Dietterich 1997]. This algorithm is similar to the FSS in the two first iterations. After the second iteration, that is, when the third candidate and the following ones are added, a backward step is performed. Here $\hat{f}_1$, $\hat{f}_2$, and $\hat{f}_3$ are considered the current set of models. Firstly, $\hat{f}_1$ is removed from the ensemble and the addition of each of the remaining candidates $\hat{f}_i (i > 3)$ to the ensemble is tested. This step is repeated for $\hat{f}_2$ and $\hat{f}_3$ and the one with the best improvement is added to the selected set. Then, further iterations are executed until a predefined number of iterations is reached.

A combined search technique that obtained good results on the feature subset selection problem, and yet has never been applied to ensemble pruning, is the floating search [Pudil et al. 1994].

4.3. Evaluation Measures for Ensemble Pruning

Evaluation measures can be used to guide the search schema and to establish the stopping criterion. Both situations are discussed here. In ranking approaches, evaluation measures are used to assess the models individually and the values obtained are used to rank them. In partitioning approaches, they are used in a similar way, but separately for each group of models. Model subset selection evaluates each of the candidate subsets as a whole.

Examples of ranking evaluation are given in Partridge and Yates [1996] and Coelho and Von Zuben [2006]. The authors rank the K_0 models according to accuracy. Then, the K most accurate models (K is a given parameter such that $K \leq K_0$) are selected. The main disadvantage of this approach is that it does not guarantee diversity, as the selected models may be very similar to each other. This problem is not so important in partitioning approaches because the process of dividing models into subgroups already ensures that models in different groups are diverse. For instance, two clustering-based partitioning approaches, ARIA [Coelho and Von Zuben 2006] and the clustering by deterministic annealing [Bakker and Heskes 2003] methods, select a single model from each cluster based on accuracy ranking.

Perrone and Cooper [1993] describe an algorithm that also ranks the models in the pool according to their accuracy. However, instead of predefining the number of models to be selected, they use an iterative procedure in order to define this number, that is, they use a different stopping criterion for the search. The authors include the candidate model in the ensemble only if the ensemble accuracy increases with the new model. Otherwise, the process stops. This method evaluates the accuracy of the base models to rank them, and evaluates the accuracy of the ensemble to decide whether the selection process should be continued or not. The description given here implies an FSS search schema. However, in Coelho and Von Zuben [2006] this evaluation procedure is integrated with both FSS and BSS search schemas under the postfix name “without exploration”.

In another approach based on accuracy, Kotsiantis and Pintelas define an implicit ranking by comparing each base model with the most accurate one using t-tests [Kotsiantis and Pintelas 2005]. Again, this approach is different because it uses a different stopping criterion.

With the exception of the clustering approaches, the approaches discussed so far [Partridge and Yates 1996; Kotsiantis and Pintelas 2005; Coelho and Von Zuben 2006] do not guarantee diversity in the searching schema. Other authors try to address this issue. Aksela selects the subset with minimal mean pair-wise correlation [Aksela 2003]. Rooney et al. use a metric that tries to balance accuracy and diversity. They define the accuracy of each model as the ratio between the generalization error of the most accurate model in the pool and each particular model. The models with an accuracy that is higher than a prespecified threshold will be selected. In a second step, the authors define the diversity for each model as the percentage of models with a correlation less than or equal to 0.6. Finally they sum the accuracy and the diversity for each model and select the K (this is a given value) models with the highest values [Rooney et al. 2004]. Rokach uses an evaluation measure that takes into account both the error of the models and the agreement on the predictions of the models [Rokach 2009a]. This pruning algorithm, known as Collective-Agreement-Based Pruning (CAP), was originally developed for classification. Although it has not been adapted for regression, this seems to be possible. An interesting approach that was developed specifically for regression is the one presented in Hernández-Lobato et al. [2006]. It evaluates each model individually using an evaluation metric that represents its bias, variance, and covariance, based on the decomposition of the error proposed in Ueda and Nakano [1996], as described in Section 2.4.

The following approaches only evaluate the performance of the ensemble. GASEN (Genetic-Algorithm-based Selective ENsemble) [Zhou et al. 2002] optimizes the performance of an ensemble that is a linear combination of all the models generated (see Section 5 for a discussion on the issue of model integration). This optimization is based on a genetic algorithm that evolves the weights of the models. However, the weights that are found in the search are not used in the integration step. Instead, GASEN prunes the ensemble by selecting those models that have a weight on the solution that is greater than a given threshold. The approach was tested with neural networks and decision trees, outperforming bagging and boosting [Zhou et al. 2002; Zhou and Tang 2003].

4.4. Discussion on Ensemble Pruning

Pruning is usually associated with methods that generate a large number of models in an independent way (e.g., Caruana et al. [2004]). This may explain why it receives less attention from the research community, which focuses more on direct approaches. Nevertheless, there is some evidence that performance improves when pruning is used with ensemble generation methods that are typically used without pruning (e.g., bagging) [Hernández-Lobato et al. 2006].

No extensive comparisons between ensemble pruning methods have been published that can provide some guidance on which method to use on a given problem. A few studies compare the method proposed by the corresponding authors with a previously proposed method [Roli et al. 2001; Partridge and Yates 1996; Coelho and Von Zuben 2006; Ruta and Gabrys 2001]. However, they all use a very small number of datasets, limiting the generality of the results.

As far as search method is concerned, a simple and yet effective search method for ensemble pruning is the FSS approach (see Section 4.2.3), described in Coelho and Von Zuben [2006] under the name *constructive with exploration*. As for more complex approaches, there are no comparative studies. Given that search-based approaches are very similar to wrapper-based feature selection, we may establish a parallel between results in both areas. The forward version of the compound method (Section 4.2.3) proposed in Pudil et al. [1994] is, according to Jain and Zongker [1997], the most effective suboptimal method on feature subset selection. Stochastic search methods, genetic algorithms [Vafaie and Jong 1993; Skalak 1994; Yang and Honavar 1997; Oh et al. 2004], simulated annealing [Loughrey and Cunningham 2005], and ant colony optimization [Al-Ani 2005] have obtained good results [Skalak 1994; Oh et al. 2004;

Table VI. Summary on Methods for Ensemble Pruning Using Partitioning-Based Approaches

Name	Stopping criterion	Evaluation measure
Clustering by deterministic annealing	Partitioning: given number of clusters	Partitioning: weighted distance (different distance measures are described)
	Selection: the best from each cluster	Selection: accuracy of the base models
Adaptive Radius Immune Algorithm (ARIA)	Partitioning: an edge will be removed if the ratio of its evaluation and the minimum evaluation of its neighbor edges is larger than a given threshold	Partitioning: Euclidean distance and local density information
	Selection: the best from each cluster	Selection: accuracy of the base models

Al-Ani 2005] on feature subset selection. Despite the parallel that can be established, it remains to be proven that these results also apply to ensemble pruning.

As far as the evaluation measure is concerned, it is advisable to use one that measures both accuracy and diversity, as the one presented in Rooney et al. [2004]. The clustering approach [Bakker and Heskes 2003; Coelho and Von Zuben 2006] can also be used for this purpose.

Pruning has been tested only with ensemble generation approaches that manipulate training data [Hernández-Lobato et al. 2006] (Section 3.1) and parameter values [Mendes-Moreira 2008] (Section 3.2.1). The combination with methods that manipulate the learning algorithm does not make sense because these methods usually control the generation process in order to ensure that an optimal ensemble is obtained. However, it would be interesting to investigate the combination of pruning with generation approaches that manipulate the model.

Additionally, it would be interesting to integrate ensemble pruning with approaches where the size of the ensemble is not determined during the generation process (i.e., typically when the number of models is defined a priori).

We believe that ensemble pruning will regain some importance in the application of ensemble methods to data streams [Wang et al. 2003; Bifet et al. 2009]. In data streams, the phenomenon that generates the data may change (referred to as *concept change* in classification). In that case, the best set of models to use in the ensemble may change. This means that some models may be kept while others become outdated and must be replaced with new ones that are generated with new data. Addressing this problem by relearning the complete ensemble will often be impractical. So, the application of ensembles to data streams will be based on a repetition of the generation and pruning steps.

As in the ensemble generation step, the majority of the methods for ensemble pruning can be used both in classification and regression.

Tables VI, VII and VIII present a summary of some pruning algorithms that can be used for ensemble regression.

5. ENSEMBLE INTEGRATION

Now that we have described the process of ensemble generation, the next step is understanding how it is possible to combine the predictions of the models in the ensemble so as to obtain a single answer. In regression problems, ensemble integration is performed using a linear combination of the predictions. This can be stated as

$$\hat{f}_{\mathcal{F}}(\mathbf{x}) = \sum_{i=1}^{K} [h_i(\mathbf{x}) * \hat{f}_i(\mathbf{x})], \tag{13}$$

where $h_i(\mathbf{x})$ are the weighting functions.

Table VII. Summary of Methods for Ensemble Pruning Using Search-Based Approaches

		Search	
Name	Approach	Stopping criterion	Evaluation measure
Pruning statistical inaccurate models	R	p-value < 0.05 in the t-test against the most accurate model	Accuracy of the base models
GASEN	R	Given threshold for the fitness	Fitness (=1/error) of the base models
50% threshold	R	Given nr. of models	Firstly by accuracy above a given threshold and then by accuracy + diversity
Correlation between errors	E	Given nr. of models	Mean pairwise correlation between errors
Forward without exploration	F	Ensemble accuracy starts decreasing	Ranking: base models' accuracy Stopping criterion: ensemble accuracy
Forward with exploration	F	Ensemble accuracy starts decreasing	Ensemble accuracy
Boosting to prune Bagging	F	Error < 0.5	Minimum weighted error
Pruning in ordered regression bagging	F	Given nr. of models	Minimum estimate of the ensemble error
Backward without exploration	B	Ensemble accuracy starts decreasing	Ranking: base models' accuracy Stopping criterion: ensemble accuracy
Backward with exploration	B	Ensemble accuracy starts decreasing	Ensemble accuracy
Greedy search with initial seed (1)	C	Last added model = last removed model	Ensemble accuracy
Reduce-error pruning with back-fitting (2)	C	Given nr. of models	Ensemble accuracy
PMEP	F	Doesn't apply	Number of validation examples that are correctly classified by more than half of the base models in the ensemble

R: Ranking; E: Exponential; F: Forward; B: Backward; C: Compound.
(1): The search schema starts with a randomized subset of a given number of models.
(2): The search schema starts with an empty subset of models.

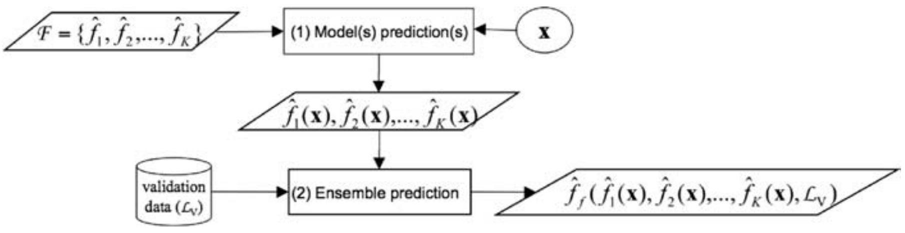
Merz divides the integration approaches into constant and nonconstant weighting functions [Merz 1998]. In the first case, the $h_i(\mathbf{x})$ are constant, that is, $h_i(\mathbf{x}) = \alpha_i$. In nonconstant weighting functions, the weights vary according to the input values $\mathbf{x}$.

When combining predictions, a possible problem is the existence of correlation between the predictions of the ensemble models, referred to as the multicollinearity problem. As a consequence, the confidence intervals for the α_i coefficients will be larger, that is, the estimators of the coefficients will have higher variance [Merz 1998]. This happens because we must determine the inverse of a linearly dependent matrix to obtain the α_i's. A common approach is to handle multicollinearity in the ensemble generation (Section 3) or in the ensemble pruning (Section 4) phases. If the principles referred to in Section 2.4 are guaranteed, namely those of accuracy and diversity, then it is possible, if not to avoid completely, at least to ameliorate this problem.

Most methods use the validation data (Section 2.3) to estimate the parameters of the weighting functions, h_i (Eq. (13)). This is the case of the constant weighting functions as defined by Merz [1998]. However, some methods use only a subset of the validation

Table VIII. Summary of Methods for Ensemble Pruning (references)

Name	Reference
Clustering by deterministic annealing	[Bakker and Heskes 2003]
ARIA: Adaptive Radius Immune Algorithm	[Coelho and Von Zuben 2006; Bezerra et al. 2005]
Pruning statistical inaccurate models	[Kotsiantis and Pintelas 2005]
GASEN	[Zhou et al. 2002]
50% threshold	[Rooney et al. 2004]
Correlation between errors	[Aksela 2003]
Forward without exploration	[Coelho and Von Zuben 2006]
Forward with exploration	[Coelho and Von Zuben 2006]
Boosting to prune bagging	[Martinez-Munoz and Suárez 2007]
Pruning in ordered regression bagging	[Hernández-Lobato et al. 2006]
Backward without exploration	[Coelho and Von Zuben 2006]
Backward with exploration	[Coelho and Von Zuben 2006]
Greedy search with initial seed	[Moreira et al. 2006]
Reduce-error pruning with back-fitting	[Margineantu and Dietterich 1997]



$\mathcal{F} = \{\hat{f}_1, \hat{f}_2, \ldots, \hat{f}_K\}$
(1) Model(s) prediction(s)
$\mathbf{x}$
$\hat{f}_1(\mathbf{x}), \hat{f}_2(\mathbf{x}), \ldots, \hat{f}_K(\mathbf{x})$
validation data ($\mathcal{L}_V$)
(2) Ensemble prediction
$\hat{f}_f(\hat{f}_1(\mathbf{x}), \hat{f}_2(\mathbf{x}), \ldots, \hat{f}_K(\mathbf{x}), \mathcal{L}_V)$

Fig. 5. Constant weighting functions model.

data in order to obtain weights that are more specialized to the given test example. This approach obviously increases the probability of overfitting.

We start this section by describing different integration functions (Section 5.1). Then we present different approaches in order to refresh $h_i(\mathbf{x})$ weights. However, the methods used for ensemble integration differ not only on how the weights are obtained, but also when and on which ensemble the data is used. All these issues are discussed in Section 5.2 and it corresponds to the dynamic approach. We also describe some comparative studies (Section 5.3) and conclude the section with a short discussion on ensemble integration.

5.1. Integration Functions

In this section, we discuss how the h_i weights are estimated, independently of whether they are estimated globally, that is, the same set of weights for all test examples, or dynamically, according to the test example. For simplification, we start by describing the integration function assuming that the entire validation set is used to define the weights. Figure 5 describes this particular situation. Nevertheless, many of the functions we describe in this section are also used in other ensemble frameworks, as described in Figure 6.

We start with simple methods that suffer the problem of multicollinearity. Then we discuss some statistical methods that address this problem, some of which are combined with search procedures.

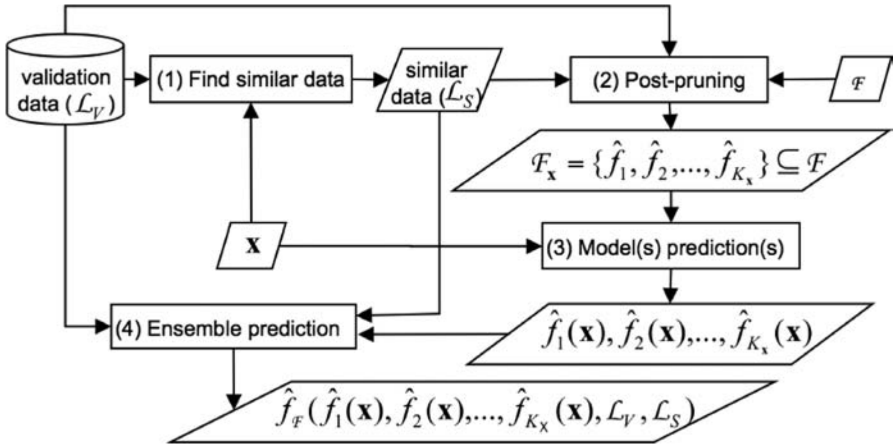

validation data ($\mathcal{L}_V$)
(1) Find similar data
similar data ($\mathcal{L}_S$)
(2) Post-pruning
$\mathcal{F}$
$\mathcal{F}_\mathbf{x} = \{\hat{f}_1, \hat{f}_2, \ldots, \hat{f}_{K_\mathbf{x}}\} \subseteq \mathcal{F}$
$\mathbf{x}$
(3) Model(s) prediction(s)
(4) Ensemble prediction
$\hat{f}_1(\mathbf{x}), \hat{f}_2(\mathbf{x}), \ldots, \hat{f}_{K_\mathbf{x}}(\mathbf{x})$
$\hat{f}_\mathcal{F}(\hat{f}_1(\mathbf{x}), \hat{f}_2(\mathbf{x}), \ldots, \hat{f}_{K_\mathbf{x}}(\mathbf{x}), \mathcal{L}_V, \mathcal{L}_S)$

Fig. 6. Dynamic approach model.

The Basic Ensemble Method (BEM) [Perrone and Cooper 1993] simply calculates the mean of the predictions of the models in the ensemble, that is, $h_i = 1/K$.

$$\hat{f}_{BEM}(\mathbf{x}) = \frac{1}{K}\sum_{i=1}^{K} \hat{f}_i(\mathbf{x}) \tag{14}$$

This method, unlike most of the others, does not depend on the models nor on the data. It assumes that the errors of the models $(f(\mathbf{x}) - \hat{f}_i(\mathbf{x}))$ are mutually independent with zero mean. The same authors proposed a more complex method, the Generalized Ensemble Method (GEM) [Perrone and Cooper 1993]. In GEM, the α_i are inversely proportional to the error in the validation set and they are also estimated taking into account the correlation between the errors of the models.

The well-known Linear Regression (LR) model is another combination method possible. The predictor is the same as in the GEM case but without the constraint $\sum_{j=1}^{K} \alpha_i = 1$. The use of a constant in the LR formula is not relevant in practice (the standard linear regression formulation uses it) because $E[\hat{f}_i(\mathbf{x})] \simeq E[f(\mathbf{x})]$ [LeBlanc and Tibshirani 1996]. It would only be necessary if predictors were meaningfully biased.

All the methods discussed so far suffer the multicollinearity problem with the exception of BEM.[1] Another method that does not suffer multicollinearity is the simple median. The use of this integration function in bagging (Section 2.5) is known as bragging [Buhlmann 2010]. Good results are reported using MARS [Friedman 1991] as base learner.

A different approach consists of methods that aim to avoid multicollinearity. Breiman presents the stacked regression [Breiman 1996c] method based on the well-known stacked generalization framework [Wolpert 1992] that was first presented in the context of classification. Given a $\mathcal{L}$ learning set with M examples, the goal is to obtain the α_i coefficients that minimize

$$\sum_{j=1}^{M}\left[f(\mathbf{x}_j) - \sum_{i=1}^{K}\alpha_i * \hat{f}_i(\mathbf{x}_j)\right]^2 \tag{15}$$

[1]However, in the case of linear regression, multicollinearity affects the coefficients but not the accuracy of the predictions [Hastie et al. 2001].

using the learning set. This raises the possibility of overfitting which can, however, be addressed by obtaining the estimates with a cross-validation process. Additionally, this method may also be affected by the multicollinearity problem. Several approaches to address this problem were investigated and it was observed that a method that consistently gives good results is the minimization of the equation presented before under the constraints $\alpha_i \geq 0, i = 1, \ldots, K$ [Breiman 1996c]. One of the methods tried out was the ridge regression method, a regression technique for solving badly conditioned problems, but results were not promising [Breiman 1996c]. An important result presented by Breiman is the empirical observation that, in most cases, many of the α_i weights are zero. This result provides further evidence that pruning (Section 4) is an important step, either as an individual phase or included in the ensemble integration phase.

Merz and Pazzani use Principal Component Regression to avoid the multicollinearity problem [Merz and Pazzani 1999]. The PCR* method obtains the Principal Components (PC), ranks them in decreasing order of the amount of variation explained, and then selects the top ones. The choice of the number of PCs is an important issue in this approach. An adequate choice is necessary to avoid underfitting or overfitting.

Wang et al. use weights that are inversely proportional to the expected error of $\hat{f}_i(\mathbf{x})$ [Wang et al. 2003]. This approach is similar to the variance-based weighting presented in Tresp and Taniguchi [1995].

Dynamic Weighting (DW) assigns a weight to each base model according to its performance [Puuronen et al. 1999; Rooney et al. 2004] and the final prediction is based on the weighted average of the predictions of the related models.

Kuncheva presents a hybrid approach that combines selection and combination approaches [Kuncheva 2002]. It uses statistical tests to verify whether one predictor is meaningfully better than the others. If positive, it uses the best predictor; if not, it uses a combination approach.

Search methods have also been applied to the problem of estimating the α_i. In a classification setting, evolutionary algorithms outperformed BEM and GEM on 25 datasets [Ortiz-Boyer et al. 2005].

Caruana et al. also follow a search approach, but embedding ensemble integration in the ensemble pruning phase [Caruana et al. 2004]. In this approach, models can be selected multiple times. Similarly to BEM, the weighting function is the simple average and so the α_i coefficients are implicitly calculated as the number of times that each model is selected over the total number of models in the ensemble (including repeated models).

A different approach from the ones described in Figure 5 is obtained by identifying regions of the input space where each model performs well. These regions are areas of expertise. Given a new example, it determines the combination of models that is expected to make the best prediction based on their areas of expertise. These methods follow a *metalearning* approach which uses machine learning algorithms to induce models that relate the characteristics of the problems with the performance of the models [Brazdil et al. 2009]. The integration function is embedded in the definition of the areas of expertise, being different for different test examples.

The work on meta decision trees [Todorovski and Dzeroski 2003] for classification applies a common algorithm for induction of decision trees on dataset containing the following independent variables: (a) the original variables, (b) characteristics of the data (e.g., mean correlation between the original variables), and (c) information about the model (e.g., the confidence it has on its prediction). The target variable of the meta tree is the model to recommend. In practice, the meta decision tree recommends a predictor rather than making a prediction directly. Although this work has been developed for classification, it is adaptable for regression by an appropriate choice of the meta attributes.

Yankov et al. use support vector machines with the Gaussian kernel to select, from an ensemble with two models, the predictor to use [Yankov et al. 2006].

5.2. Dynamic Approach

In the dynamic approach, the selection of predictors is done on-the-fly. Given a new example, it chooses the predictors that are expected to make the best combined prediction. While in the approach by areas of expertise the region of the input space where each model is expected to perform best is previously defined, in the dynamic approach the areas are defined on-the-fly. Sometimes, in the dynamic approach only a subset of the ensemble is used for prediction, avoiding the use of models potentially inaccurate for a given test example. This selection is referred to as postpruning.

Figure 6 summarizes the dynamic approach. Given an input vector $\mathbf{x}$, it first selects similar data. Then, according to the performance of the models on similar data, a number $K_{\mathbf{x}}$ of models are selected from the ensemble $\mathcal{F}$ (Eq. (10)). Merz describes this approach in detail, namely the use of a performance matrix to evaluate the models locally [Merz 1996]. It consists of a $M \times K$ matrix, where M is the number of training examples and K is the number of models in $\mathcal{F}$. The matrix contains errors of the models on the training examples. In regression, the error measure can be, for instance, the squared error, the absolute error, or other performance measure. If $K_{\mathbf{x}} = 1$, usually known as the dynamic selection or adaptive selection approach [Giacinto and Roli 1997; Woods 1997; Kuncheva 2002], then the prediction of the ensemble is obviously the prediction of the selected model. If $K_{\mathbf{x}} > 1$, known as the dynamic combination or (model) fusion approach [Woods 1997], then the integration method uses the performances of similar data (sd) obtained from the validation data (vd) to estimate the $h_i(\mathbf{x})$ weights.

This approach consists of the following steps (assuming that the ensemble models are already available):

(1) Given an input value $\mathbf{x}$, find a similar dataset ($\mathcal{L}_S$) from the validation set ($\mathcal{L}_V$), such that $\mathcal{L}_S \subseteq \mathcal{L}_V$.
(2) Select a model's subset $\mathcal{F}_{\mathbf{x}} \subseteq \mathcal{F}$ from the ensemble according to their performance for the selected similar data $\mathcal{L}_S$, that is, postpruning.
(3) Obtain the prediction $\hat{f}_i(\mathbf{x})$ for the given input value, for each selected $\hat{f}_i \in \mathcal{F}_{\mathbf{x}}$.
(4) Obtain the ensemble prediction $\hat{f}_{\mathcal{F}_{\mathbf{x}}}$. This is straightforward if just one model is selected; otherwise, it is necessary to combine results using, typically, one of the integration functions discussed in Section 5.1.

While step (3) is straightforward, the others are not. In this section, related works concerning the remaining three steps are reviewed.

The standard method for obtaining similar data (step 1) in the context of ensemble learning is the well-known k-nearest neighbors with the Euclidean distance [Woods 1997]. The Heterogeneous Euclidean Overlap Measure [Wilson and Martinez 1997] is used when there are symbolic input variables [Tsymbal et al. 2006a]. One limitation of these methods is that they weigh equally all the input variables even if there are input variables with different levels of relevance in the explanation of the target variable. Some authors measure similarities using attribute weighted metrics in the context of random forests [Robnik-Šikonja 2004; Tsymbal et al. 2006b] obtaining increased accuracy comparatively to the standard integration method of random forest, the simple average. Didaci and Giacinto also test a kind of similarity measure according to the outputs [Didaci and Giacinto 2004] embedded in DANN - Discriminant Adaptive Nearest Neighbor [Hastie and Tibshirani 1996]. DANN locally reshapes the nearest neighborhood. In practice, some of the explanatory variables are discarded, reducing the dimensionality of the problem. Experiments performed by Didaci and Giacinto show that this approach, as well as a dynamic choice of the number k of neighbors, can meaningfully improve the results when compared with the standard Euclidean distance [Didaci and Giacinto 2004].

The simplest postpruning method (step 2) is choosing the one with the best performance according to a given metric [Woods 1997; Giacinto and Roli 1997]. However, the dynamic selection approach can use more than one model [Merz 1996]. The dynamic weighting with selection [Rooney et al. 2004] uses the 50% more accurate models locally from the ensemble.

When more than one model is selected, their results are combined (step 4). This is typically done with a linear function as in Eq. (13), where $h_i(\mathbf{x}) = \alpha_{\mathbf{x},i}$ is the weight of model $\hat{f}_i$ specifically for example $\mathbf{x}$. This subject has already been discussed in Section 5.1. Furthermore, we have also described some of the integration functions already used in the dynamic framework [Rooney et al. 2004; Wang et al. 2003].

5.3. Comparative Studies

The main study comparing constant weighting functions in regression is presented by Merz [1998]. The functions used are: GEM, BEM, LR, LRC (the LR formula with a constant term), gradient descent, EG, EG^{+}_{-} (the last three methods are gradient descent procedures discussed in Kivinen and Warmuth [1997]), ridge regression, constrained regression (Merz [1998] uses the bounded variable least squares method from Stark and Parker [1995]), stacked constrained regression (with ten partitions), and PCR*. In one experiment, an ensemble of 12 models on 8 regression datasets were used: six of the models were generated using MARS [Friedman 1991] and the other six using the neural network back-propagation algorithm. The three globally best functions were constrained regression, EG, and PCR*. The other experiment tested how the functions perform with many correlated models. The author used neural network ensembles size ten and fifty. Just three datasets were used. The PCR* function presented more robust results.

The main study on the dynamic approach is the one performed by Mendes-Moreira et al. [2009]. They present two main studies, one on methods for the selection of similar data, and another on postpruning methods and (nonconstant) integration functions.

The study on methods for the selection of similar data compares the algorithm k-nearest neighbors using different distance measures, namely, the Euclidean, kd-tree [Bentley 1975], and RReliefF measures [Robnik-Šikonja and Kononenko 2003], and all examples that fall together with the test instance in the same leaf node of a decision tree. The use of k-nearest neighbors with kd-tree and RReliefF measures presents globally the best results in five regression datasets. The optimal number of nearest neighbors is problem-dependent. In the CART approach, the number of similar examples depends on the size of the leaf node where the test example falls. This fact can influence the comparison between the CART approach and all the others. This experience shows that the best distance measures are those that weigh the features according to what extent the target variable can be explained.

The study on postpruning methods and integration functions compares DW and DWS with different settings, as well as the selection of the best model and forward selection with replacement [Caruana et al. 2004]. Results on five regression datasets confirm the results presented in Rooney et al. [2004], that is, the best results are obtained using DWS.

5.4. Discussion on Ensemble Integration

The comparative study published by Merz in 1998 [Merz 1998] is the most complete source of information concerning the selection of ensemble integration methods.

However, since 1995, most research has been focusing on the ensemble generation step. Advances in the studies on the generalization ensemble error (Section 2.4) show that an important part of the problems that arise in the integration phase can be solved by a joint design of the three phases (generation, pruning, and integration) [Rosen 1996; Liu et al. 2000; Zhou et al. 2002; Rodríguez et al. 2006]. Constant weighting functions were used on all the studies mentioned in Section 3.3. The reason for this is that the decomposition of the generalization error (Section 2.4) for the simple average is known. This enables the development of ensemble generation methods that minimize the error. The approach seems to have changed from "which integration function to use for a given ensemble?" to "how to generate the ensemble for a given integration function?".

The main disadvantage of constant weighting functions is that the α_i weights, being equal for the entire input space, can, at least theoretically, be less adequate for some parts of the input space. This is the main argument for using nonconstant weighting functions [Verikas et al. 1999]. Little effort has been devoted to the problem of how to generate the best ensemble for a given nonconstant weighting function (e.g., Mendes-Moreira [2008]). This is due to the difficulty of decomposing the error when using this kind of integration functions. Nevertheless, we believe that this is an interesting topic for future research.

As far as dynamic integration methods are concerned, the question of which is the best approach, as well as the selection of a single model versus the combination of multiple models, were debated for a long time [Kuncheva 2002]. Recently, it has been shown that there is no advantage in the former over the combination approach [Ko et al. 2008; Mendes-Moreira et al. 2009]. The right question for the dynamic approach is on the choice of the nonconstant integration function to use, as discussed previously.

The dynamic approach is reported to give good results in time changing phenomena [Wang et al. 2003; Kolter and Maloof 2007; Tsymbal et al. 2008]. This makes it particularly suitable for data streams, for the same reasons that make pruning an important step in this kind of data (Section 4.4) [Wang et al. 2003; Bifet et al. 2009]. In fact, pruning can be regarded as a particular case of model integration, where eliminated models are assigned a null weight.

As observed previously, dynamic approaches are essentially mapping (characteristics of the) data for model performance. This is the type of problem that is addressed in metalearning [Brazdil et al. 2009]. However, little work has been carried out on the use of this approach for ensemble integration, which is another interesting topic for future research.

Ensemble integration is the step where the biggest differences between classification and regression can be identified. This is due to the difference in the nature of the outputs. For instance, majority voting, which is often used in classification, cannot be used in regression. On the other hand, mean values cannot be used in the integration of models for classification tasks.

Tables IX and X present a summary of some integration methods that can be used for ensemble regression.

6. GENERAL DISCUSSION

Several empirical studies confirm the advantage of using ensembles over single models. For instance, random forests are consistently among the best three models in the benchmark study carried out by Meyer et al. [2003], which included many different algorithms.

To decide whether ensemble methods should be considered for a given regression problem, an important criterion is the amount of data available. If data are plentiful, then they should be divided into training, validation, and test sets (Section 2.3). Otherwise, data may not be sufficient for obtaining reliable models and/or assessing their

Table IX. Summary on Methods for Ensemble Integration

Ref.	Name	Integration function	Dynamic?
[Perrone and Cooper 1993]	Basic ensemble method	SA	No
[Perrone and Cooper 1993]	Generalized ensemble method	WA	No
[Breiman 1996c]	Linear regression model	WA	No
[Merz and Pazzani 1999]	PCR*	WA	No
[Caruana et al. 2004]	Forward selection with replacement	WA	No
[Todorovski and Dzeroski 2003]	Meta decision trees	S	No
[Rooney et al. 2004]	Dynamic selection	S	Yes
[Rooney et al. 2004]	Dynamic weighting	WA	Yes
[Rooney et al. 2004]	Dynamic weighting with selection	S & WA	Yes

SA: Simple Average; WA: Weighted Average; S: Selection.

Table X. Summary on Methods for Ensemble Integration (continuation)

Name	Data used for weight estimation	Description
Basic ensemble method	Validation set	It weighs all models equally, i.e., the simple average
Generalized ensemble method	Validation set	The weighs, which sum is 1, are inversely proportional to the error in the validation set
Linear regression model	Validation set	The weighs are inversely proportional to the error in the validation set but there is no constraint on the sum of the weights
PCR*	Validation set	It selects the top models (components) according to how much variation they explain under principal component analysis
Forward selection with replacement	Validation set	It selects with reposition the model that increases the ensemble accuracy in a validation set and averages results
Meta decision trees	Validation set	It uses a tree where the nodes are meta-characteristics of the models to recommend the model to use
Dynamic selection	k-nn	It selects the model with less error on the k-nearest neighbors set
Dynamic weighting	k-nn	The weights are inversely proportional to the error in neighbors set
Dynamic weighting with selection	k-nn	It is like dynamic weighting but discards the models with an error higher than a pre-defined threshold by comparison against the most accurate model

quality in a reliable way. One possibility is not to create a separate validation set and use the training set for internal validation during the ensemble learning process. In this case, overfitting can occur.

Availability is also an important criterion when deciding whether to use ensemble methods or not. The successful results of these methods have led data mining tool vendors to incorporate them into their tools. For example, SAS Enterprise Miner implements bagging, boosting, and heterogeneous ensembles [Matignon 2007]. Another leading tool that claims to use ensemble methods is SPSS Modeler. Open-source tools usually focus more on the methods and, thus, typically have a larger offer of ensemble methods, for both regression and classification. Tools such as RapidMiner, WEKA [Witten and Frank 2011], and R implement not only the most common methods like bagging and boosting, but also more advanced ones, such as stacking, random forest, and rotation forest.

An advantage of most ensemble methods is their ease of use. Typically they have few parameters and are reasonably robust to their values.

As far as computational complexity is concerned, ensemble methods should not be expected to be the fastest methods as they require, by definition, multiple executions of one or more algorithms. The problem becomes particularly important in methods that are based on search, either in the generation step 3 or in the pruning step 4. These methods typically require the evaluation of multiple solutions (i.e., multiple executions of the learning algorithm). For some of the methods, such as bagging or random forest, this problem may be easily addressed by parallelization because each model of the ensemble is generated independently from the others. The scalability regarding the number of both examples and features is not determined by the ensemble method but by the base-level learning algorithm.

There is not much information available on the choice of method. As shown in this article, many alternative methods can be used at each step of the ensemble learning process. However, there is very little knowledge about the strengths and weaknesses of each method. In fact, the results reported in different papers are not comparable because of the use of different experimental setups [Islam et al. 2003; García-Pedrajas et al. 2005]. A contribution that would be of major importance to the field is the thorough empirical investigation of each approach and the characterization of the conditions under which each one should and should not be used. An approach to carrying empirical studies that would be useful in this case is called experiment databases [Blockeel and Vanschoren 2007].

As for future work, one approach that can be explored and seems to be promising but has not been discussed earlier, because it does not pertain to one particular step, is to combine different ensemble integration methods. The method wMetaComb [Rooney and Patterson 2007] uses a weighted average to combine stacked regression (described in Section 5.1) and the DWS dynamic method (Section 5.2). The cocktail ensemble for regression [Yu et al. 2007] combines different ensemble approaches, whichever they are, using a combination derived from the ambiguity decomposition. It combines different ensembles using forward selection to choose the one that reduces the combined estimated error the most. The same ensemble can be selected more than once.

7. CONCLUSIONS

Ensemble learning is concerned with methods that combine several models to make predictions. The main advantage of ensemble methods is their accuracy and robustness comparatively to the use of a single model.

For ensemble learning, as for other research areas, methods for regression and for classification are based on different solutions, at least partially. This is particularly true in approaches that manipulate the induction algorithm in the ensemble generation phase and in the ensemble integration phase, which depends on the type of output to be combined. This has caused a significant amount of work to be carried out independently on ensembles for regression. However, despite the importance of regression, the only surveys on ensemble learning that are available to both researchers and practitioners focus on classification. This article tries to address this problem by presenting a review of ensemble learning for regression. Nevertheless, we believe that this article can also be useful to the field of ensembles for classification. Besides the methods that are independent of the learning task, the taxonomy that is proposed to organize the methods is also applicable to ensembles of classifiers.

Ensemble learning is typically divided into three phases: generation, pruning, and integration. The generation phase aims to obtain an ensemble of models. It can be

Table XI. Main Homogeneous Ensemble Generation Methods for Regression

Method	Reference	Algorithm	Class/Regr
Bagging	[Breiman 1996a]	Unst. learners	yes / yes
Random forests	[Breiman 2001a]	Decis. trees	yes / yes
EENCL	[Liu et al. 2000]	ANN	yes / yes
CNNE	[Islam et al. 2003]	ANN	yes / yes
Coop. Coev.	[García-Pedrajas et al. 2005]	ANN	yes / ?

classified as homogeneous or as heterogeneous depending on the number of induction algorithms used. The most successful methods for ensemble generation are developed for unstable learners, that is, learners that are sensitive to changes in the training set, namely decision trees or neural networks. Table XI summarizes some of the most important methods on homogeneous ensemble generation. The "?" symbol means that this method has not been tested for regression, and consequently it is not known how it would work for regression.

Ensemble pruning selects a subset from a pool of models to reduce computational complexity and, if possible, to increase accuracy. It has many similarities with the well-known feature subset selection task. This happens because in both cases the goal is to select a subset from a set of objects in order to optimize a given objective function. As in the feature subset selection case, randomized heuristics, such as evolutionary algorithms or tabu search, seem very effective. Despite the recent reduction of importance of the direct ensemble generation methods (without a pruning step), research on pruning has meanwhile been done by addressing generating approaches initially designed to be direct.

Ensemble integration functions use the predictions made by the models in the ensemble to obtain the final ensemble prediction. They can be classified as constant or nonconstant weighting functions. As previously underlined, constant weighting functions are the most frequently used, possibly because it is easier to generate ensembles in order to minimize known generalization error functions in regression. However, since nonconstant weighting functions seem to be attractive in order to increase accuracy, further research is needed to obtain ensemble methods that take advantage of such integration functions.

This article describes the complete process for ensemble-based regression and discusses each phase thoroughly. As shown previously, there are many alternative methods that can be used for each step. However, very little guidance is available for practitioners to select which ensemble method should be used on a given regression problem. This work also identifies many challenging problems to be solved at each step and many ideas that still need theoretical and experimental development.

The number of papers published on ensemble regression is too large to be exhaustively discussed in a single survey. Furthermore, many papers represent small variants of previously proposed approaches. Here, we have identified the main trends and described some of the most representative papers for each of them. We believe that this work provides a thorough road map that can serve as a stepping stone to new research ideas as well as provide support for practitioners to choose the most appropriate solution for their regression applications.

REFERENCES

Aggarwal, N., Prakash, N., and Sofat, S. 2010. Content management system effort estimation using bagging predictors. In *Proceedings of the International Joint Conference on Computer Information Systems Sciences and Engineering Technological Developments in Education and Automation*, M. Iskander, V. Kapila, and M. Karim, Eds. 19–24.

AHA, D. W. AND BANKERT, R. L. 1996. A comparative evaluation of sequential feature selection algorithms. In *Learning from Data*, D. Fisher and H.-J. Lenz, Eds. Springer, Chapter 4, 199–206.

AKSELA, M. 2003. Comparison of classifier selection methods for improving committee performance. In *Proceedings of the International Workshop on Multiple Classifier Systems.* Lecture Notes in Computer Science, vol. 2709. Springer, 84–93.

AL-ANI, A. 2005. Feature subset selection using ant colony optimization. *Int. J. Comput. Intell. 2,* 1, 53–58.

AVNIMELECH, R. AND INTRATOR, N. 1999. Boosting regression estimators. *Neural Comput. 11*, 499–520.

AZEVEDO, P. J. AND JORGE, A. M. 2007. Iterative reordering of rules for building ensembles without relearning. In *Proceedings of the International Conference on Discovery Science.* Lecture Notes in Computer Science, vol. 4755. Springer, 56–67.

AZEVEDO, P. J. AND JORGE, A. M. 2010. Ensembles of jittered association rule classifiers. *Data Min. Knowl. Discov. 21*, 1, 91–129.

BAKKER, B. AND HESKES, T. 2003. Clustering ensembles of neural network models. *Neural Netw. 16*, 2, 261–269.

BENTLEY, J. L. 1975. Multidimensional binary search trees used for associative searching. *Comm. ACM 18*, 9, 509–517.

BEZERRA, G. B., BARRA, T. V., CASTRO, L. N., AND VON ZUBEN, F. J. 2005. Adaptive radius immune algorithm for data clustering. In *Proceedings of the International Conference on Artificial Immune Systems (ICARIS'05).* Lecture Notes in Computer Science, vol. 3627. Springer, 290–303.

BIFET, A., HOLMES, G., PFAHRINGER, B., KIRKBY, R., AND GAVALDÀ, R. 2009. New ensemble methods for evolving data streams. In *Proceedings of the Annual ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'09).* ACM, New York, 139–148.

BLOCKEEL, H. AND VANSCHOREN, J. 2007. Experiment databases: Towards an improved experimental methodology in machine learning. In *Proceedings of the 11th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'07)*. Lecture Notes in Computer Science, vol. 4702. Springer, 6–17.

BORRA, S. AND CIACCIO, A. D. 2002. Improving nonparametric regression methods by bagging and boosting. *Comput. Statist. Data Anal. 38*, 4, 407–420.

BRAZDIL, P., GIRAUD-CARRIER, C., SOARES, C., AND VILALTA, R. 2009. *Metalearning: Applications to Data Mining*. Springer.

BREIMAN, L. 1996a. Bagging predictors. *Mach. Learn. 26*, 123–140.

BREIMAN, L. 1996b. Heuristics of instability and stabilization in model selection. *Ann. Statist. 24*, 6, 2350–2383.

BREIMAN, L. 1996c. Stacked regressions. *Mach. Learn. 24,* 49–64.

BREIMAN, L. 2000. Randomizing outputs to increase prediction accuracy. *Mach. Learn. 40*, 3, 229–242.

BREIMAN, L. 2001a. Random forests. *Mach. Learn. 45*, 5–32.

BREIMAN, L. 2001b. Using iterated bagging to debias regressions. *Mach. Learn. 45*, 3, 261–277.

BROWN, G. 2004. Diversity in neural network ensembles. Ph.D. thesis, University of Birmingham.

BROWN, G., WYATT, J. L., HARRIS, R., AND YAO, X. 2005a. Diversity creation methods: A survey and categorisation. *Inf. Fusion 6*, 5–20.

BROWN, G., WYATT, J. L., AND TINO, P. 2005b. Managing diversity in regression ensembles. *J. Mach. Learn. Res. 6*, 1621–1650.

BUHLMANN, P. 2010. *Bagging, Boosting and Ensemble Methods*. Springer.

BUJA, A. AND STUETZLE, W. 2006. Observations on bagging. *Statistica Sinica 16*, 323–351.

CAI, T. AND WU, X. 2008. Research on ensemble learning based on discretization method. In *Proceedings of the 9th International Conference on Signal Processing (ICSP'08).* 1528–1531.

CARUANA, R., NICULESCU-MOZIL, A., CREW, G., AND KSIKES, A. 2004. Ensemble selection from libraries of models. In *International Conference on Machine Learning.*

COELHO, G. P. AND VON ZUBEN, F. J. 2006. The influence of the pool of candidates on the performance of selection and combination techniques in ensembles. In *Proceedings of the International Joint Conference on Neural Networks.* 10588–10595.

DELVE. 2002. Delve: Data for evaluating learning in valid experiments. http://www.cs.toronto.edu/∼delve/

DIDACI. L. AND GIACINTO, G. 2004. Dynamic classifier selection by adaptive k-nearest neighbourhood rule. In *Proceedings of the International Workshop on Multiple Classifier Systems*, F. Roli, J. Kittler, and T. Windeatt, Eds. Lecture Notes in Computer Science, vol. 3077. Springer, 174–183.

DIETTERICH, T. G. 1997. Machine-Learning research: Four current directions. *AI Mag. 18*, 4, 97–136.

DOMENICONI, C. AND YAN, B. 2004. Nearest neighbor ensemble. In *Proceedings of the International Conference on Pattern Recognition.* Vol. 1. 228–231.

DOMINGOS, P. 1997. Why does bagging work? A Bayesian account and its implications. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining.* AAAI Press, 155–158.

DRUCKER, H. 1997. Improving regressors using boosting techniques. In *Proceedings of the 14th International Conference on Machine Learning (ICML'97).* Morgan Kaufmann Publishers, San Fransisco, CA, 107–115.

DUFFY, N. AND HELMBOLD, D. 2002. Boosting methods for regression. *Mach. Learn. 47*, 153–200.

FERRER, L., SÖNMEZ, K., AND SHRIBERG, E. 2009. An anticorrelation kernel for subsystem training in multiple classifier systems. *J. Mach. Learn. Res. 10*, 2079–2114.

FLANNAGAN, S. AND SPERBER, L. 2008. Datamob/datasets. http://datamob.org/datsets

FRANK, A. AND ASUNCION, A. 2010. UCI machine learning repository. http://archive.ics.uci.edu/ml

FRANK, E. AND PFAHRINGER, B. 2006. Improving on bagging with input smearing. In *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 97–106.

FREUND, Y. AND SCHAPIRE, R. 1996. Experiments with a new boosting algorithm. In *Proceedings of the International Conference on Machine Learning*. 148–156.

FREUND, Y. AND SCHAPIRE, R. E. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci. 55*, 119–139.

FRIEDMAN, J. H. 1991. Multivariate adaptive regression splines. *The Ann. Statist. 19*, 1, 1–141.

FRIEDMAN, J. H. 1996. Local learning based on recursive covering. Tech. rep.

FRIEDMAN, J. H. 2001. Greedy function approximation: A gradient boosting machine. *Ann. Statist. 29*, 5, 1189–1232.

FRIEDMAN, J. H. 2002. Stochastic gradient boosting. *Comput. Statist. Data Anal. 38*, 4, 367–378.

FRIEDMAN, H. H. AND STUETZLE, W. 1981. Projection pursuit regression. *J. Amer. Statist. Regress. 76*, 376, 817–823.

GARCÍA-PEDRAJAS, N., HERVÁS-MARTÍNEZ, C., AND ORTIZ-BOYER, D. 2005. Cooperative coevolution of artificial neural network ensembles for pattern classification. *IEEE Trans. Evolut. Comput. 9*, 3, 271–302.

GEMAN, S., BIENENSTOCK, E., AND DOURSAT, R. 1992. Neural networks and the bias/variance dilemma. *Neural Comput. 4*, 1, 1–58.

GIACINTO, G. AND ROLI, F. 1997. Adaptive selection of image classifiers. In *Proceedings of the International Conference on Image Analysis and Processing.* Lecture Notes in Computer Science, vol. 1310. Springer, 38–45.

GRANITTO, P., VERDES, P., AND CECCATTO, H. 2005. Neural network ensembles: Evaluation of aggregation algorithms. *Artif. Intell. 163*, 2, 139–162.

GUVENIR, H. A, AND UYSAL, I. 2000. Function approximation repository. http://funapp.cs.bilkent.edu.tr/DataSets/

HASHEM, S. 1993. Optimal linear combinations of neural networks. Ph.D. thesis, Purdue University.

HASTIE, T. AND TIBSHIRANI, R. 1996. Discriminant adaptive nearest neighbor classification. *IEEE Trans. Pattern Anal. Mach. Intell. 18*, 6, 607–616.

HASTIE, T., TIBSHIRANI, R., AND FRIEDMAN, J. H. 2001. *The Elements of Statistical Learning: Data Mining, Inference, and Predictions*. Springer Series in Statistics. Springer.

HERNÁNDEZ-LOBATO, D., MARTÍNEZ-MUÑOZ, G., AND SUÁREZ, A. 2006. Pruning in ordered regression bagging ensembles. In *Proceedings of the International Joint Conference on Neural Networks*. 1266–1273.

HO, T. K. 1998. The random subspace method for constructing decision forests. *IEEE Trans. Pattern Anal. Mach. Intell. 20*, 8, 832–844.

ISLAM, M. M., YAO, X., AND MURASE, K. 2003. A constructive algorithm for training cooperative neural network ensembles. *IEEE Trans. Neural Netw. 14*, 4, 820–834.

JAIN, A. AND ZONGKER, D. 1997. Feature selection: Evaluation, application, and small sample performance. *IEEE Trans. Pattern Anal. Mach. Intell. 19,* 2, 153–158.

JORGE, A. M. AND AZEVEDO, P. J. 2005. An experiment with association rules and classification: Post-Bagging and conviction. In *Discovery Science*. Lecture Notes in Computer Science, vol. 3735. Springer, 137–149.

KIM, H.-C., PANG, S., JE, H.-M., KIM, D., AND BANG, S.-Y. 2003. Constructing support vector machine ensemble. *Pattern Recogn. 36*, 12, 2757–2767.

KIVINEN, J. AND WARMUTH, M. K. 1997. Exponentiated gradient versus gradient descent for linear predictors. *Inf. Comput. 132*, 1, 1–63.

KO, A. H.-R., SABOURIN, R., AND BRITTO JR., A. D. S. 2008. From dynamic classifier selection to dynamic ensemble selection. *Pattern Recogn. 41*, 1718–1731.

KOLEN, J. F. AND POLLACK, J. B. 1990. Back propagation is sensitive to initial conditions. Tech. rep. TR-90-JK-BPSIC, The Ohio State University.

KOLTER, J. Z. AND MALOOF, M. A. 2007. Dynamic weighted majority: An ensemble method for drifting concepts. *J. Mach. Learn. Res. 8*, 2755–2790.

KOTSIANTIS, S. AND PINTELAS, P. 2005. Selective averaging of regression models. *Ann. Math. Comput. Teleinf. 1*, 3, 65–74.

KROGH, A. AND VEDELSBY, J. 1995. Neural network ensembles, cross validation, and active learning. *Adv. Neural Inf. Process. Syst. 7,* 231–238.

KUNCHEVA, L. I. 2002. Switching between selection and fusion in combining classifiers: An experiment. *IEEE Trans. Syst. Man Cybernet. B32*, 2, 146–156.

KUNCHEVA, L. I. 2004. *Combining Pattern Classifiers*. Wiley.

LASOTA, T., TELEC, Z., TRAWINSKI, B., AND TRAWINSKY, K. 2009. A multi-agent system to assist with real estate appraisals using bagging ensembles. In *Computational Collective Intelligence: Semantic Web, Social Networks and Multiagent Systems*, N. NGUYEN, R. KOWALCZYK, AND S. CHEN, Eds. Lecture Notes in Artificial Intelligence, vol. 5796. Springer, 813–824.

LAZAREVIC, A. 2001. Effective pruning of neural network classifier ensembles. In *Proceedings of the International Joint Conference on Neural Networks*. 796–801.

LEBLANC, M. AND TIBSHIRANI, R. 1996. Combining estimates in regression and classification. *J. Amer. Statist. Assoc. 91*, 1641–1650.

LEHMANN, E. 1998. *Theory of Point Estimation*. Springer.

LIN, H.-T. AND LI, L. 2005. Infinite ensemble learning with support vector machines. In *Proceedings of the European Conference on Machine Learning*. Lecture Notes in Artificial Intelligence, vol. 3720. Springer, 242–254.

LIU, Y. AND YAO, X. 1999. Ensemble learning via negative correlation. *Neural Netw. 12*, 1399–1404.

LIU, Y., YAO, X., AND HIGUCHI, T. 2000. Evolutionary ensembles with negative correlation learning. *IEEE Trans. Evolut. Comput. 4*, 4, 380–387.

LOUGHREY, J. AND CUNNINGHAM, P. 2005. Using early stopping to reduce overfitting in wrapper-based feature weighting. Tech. rep. TCD-CS-2005-41, Trinity College Dublin, Computer Science Department.

MARGINEANTU, D. D. AND DIETTERICH, T. G. 1997. Pruning adaptive boosting. In *Proceedings of the International Conference on Machine Learning*. 211–218.

MARTINEZ-MUNOZ, G., HERNANDEZ-LOBATO, D., AND SUAREZ, A. 2009. An analysis of ensemble pruning techniques based on ordered aggregation. *IEEE Trans. Pattern Anal. Mach. Intell. 31*, 245–259.

MARTINEZ-MUNOZ, G. AND SUAREZ, A. 2006. Pruning in ordered bagging ensembles. In *Proceedings of the International Conference on Machine Learning*. 609–616.

MARTINEZ-MUNOZ, G. AND SUAREZ, A. 2007. Using boosting to prune bagging ensembles. *Pattern Recogn. Lett. 28*, 1, 156–165.

MATIGNON, R. 2007. *Data Mining Using SAS Enterprise Miner*. Wiley.

MENDES-MOREIRA, J. 2008. Travel time prediction for the planning of mass transit companies: A machine learning approach. Ph.D. thesis, Faculty of Engineering, University of Porto.

MENDES-MOREIRA, J., JORGE, A. M., FREIRE DE SOUSA, J., AND SOARES, C. 2012. Comparing state-of-the-art regression methods for long term travel time prediction. *Intell. Data Anal. 16*, 3.

MENDES-MOREIRA, J., JORGE, A. M., SOARES, C., AND FREIRE DE SOUSA, J. 2009. Ensemble learning: A study on different variants of the dynamic selection approach. In *Proceedings of the 6th International Conference on Machine Learning and Data Mining*, P. Perner, Ed. Lecture Notes in Computer Science, vol. 5632. Springer, 191–205.

MENG, G., PAN, C., ZHENG, N.,, AND SUN, C. 2010. Skew estimation of document images using bagging. *IEEE Trans. Image Process. 19*, 7, 1837–1846.

MERZ, C. J. 1996. Dynamical selection of learning algorithms. In *Proceedings of the International Workshop on Artificial Intelligence and Statistics,* D. Fisher and H.-J. Lenz, Eds. Springer.

MERZ, C. J. 1998. Classification and regression by combining models. Ph.D. thesis, University of California Irvine.

MERZ, C. J. AND PAZZANI, M. J. 1999. A principal components approach to combining regression estimates. *Mach. Learn. 36*, 9–32.

MEYER, D., LEISCH, F., AND HORNIK, K. 2003. The support vector machine under test. *Neurocomput. 55*, 1–2, 169–186.

MLG, U. D. 2011. http://mlg.ucd.ie/

Molina, L. C., Belanche, L., and Nebot, A. 2002. Feature selection algorithms: A survey and experimental evaluation. In *Proceedings of the IEEE International Conference on Data Mining.* 306–313.

Monti, S., Tamayo, P., Mesirov, J., and Golub, T. 2003. Consensus clustering: A resampling-based method for class discovery and visualization of gene expression microarray data. *Mach. Learn.*, 91–118.

Moreira, J. M., Sousa, J. F., Jorge, A. M., and Soares, C. 2006. An ensemble regression approach for bus trip time prediction. In *Proceedings of the Meeting of the EURO Working Group on Transportation.* 317–321. http://www.liaad.up.pt/~amjorge/docs/Triana/Moreira06b.pdf.

Oh, I.-S., Lee, J.-S., and Moon, B.-R. 2004. Hybrid genetic algorithms for feature selection. *IEEE Trans. Pattern Anal. Mach. Intell. 26*, 11, 1424–1437.

Opitz, D. W. 1999. Feature selection for ensembles. In *Proceedings of the National Conference on Artificial Intelligence.* AAAI Press, 379–384.

Opitz, D. W. and Shavlik, J. W. 1996. Generating accurate and diverse members of a neural-network ensemble. *Adv. Neural Inf. Process. Syst. 8*, 535–541.

Ortiz-Boyer, D., Hervas-Martinez, C., and Garcia-Pendrajas, N. 2005. Cixl2: A crossover operator for evolutionary algorithms based on population features. *J. Artif. Intell. Res. 24*, 1–48.

Parmanto, B., Munro, P, W., and Doyle, H. R. 1996. Reducing variance of committee prediction with resampling techniques. *Connect. Sci. 8,* 3–4, 405–425.

Partridge, D. and Yates, W. B. 1996. Engineering multiversion neural-net systems. *Neural Comput. 8*, 4, 869–893.

Perrone, M. P. and Cooper, L. N. 1993. When networks disagree: Ensemble methods for hybrid neural networks. In *Neural Networks for Speech and Image Processing*, R. Mammone, Ed. Chapman-Hall.

Polikar, R. 2009. Ensemble learning. *Scholarpedia 4*, 1, 2776.

Pudil, P., Ferri, F., Novovicova, J., and Kittler, J. 1994. Floating search methods for feature selection with nonmonotonic criterion functions. In *Proceedings of the IEEE International Conference on Pattern Recognition.* Vol. 11. 279–283.

Puuronen, S., Terziyan, V., and Tsymbal, A. 1999. A dynamic integration algorithm for an ensemble of classifiers. In *Proceedings of the International Symposium on Methodologies for Intelligent Systems.* Lecture Notes in Computer Science, vol. 1609. Springer, 592–600.

Ranawana, R. and Palade, V. 2006. Multi-Classifier systems: Review and a roadmap for developers. *Int. J. Hybrid Intell. Syst. 3*, 1, 35–61.

Ratsch, G., Demiriz, A., and Bennett, K. P. 2002. Sparse regression ensembles in infinite and finite hypothesis spaces. *Mach. Learn. 48*, 189–218.

Raviv, Y. and Intrator, N. 1996. Bootstrapping with noise: An effective regularization technique. *Connect. Sci. 8*, 3–4, 355–372.

Robnik-Sikonja, M. 2004. Improving random forests. In *Proceedings of the European Conference on Machine Learning.* Lecture Notes in Artificial Intelligence, vol. 3201. Springer, 359–370.

Robnik-Sikonja, M. and Kononenko, I. 2003. Theoretical and empirical analysis of relieff and rrelieff. *Mach. Learn. 53*, 1-2, 23–69.

Rodriguez, J. J., Kuncheva, L. I., and Alonso, C. J. 2006. Rotation forest: A new classifier ensemble. *IEEE Trans. Pattern Anal. Mach. Intell. 28*, 10, 1619–1630.

Rokach, L. 2009a. Collective-Agreement-Based pruning of ensembles. *Comput. Statist. Data Anal. 53,* 1015–1026.

Rokach, L. 2009b. *Pattern Classification Using Ensemble Methods*. World Scientific.

Rokach, L. 2009c. Taxonomy for characterizing ensemble methods in classification tasks: A review and annotated bibliography. *Comput. Statist. Data Anal. 53*, 12, 4046–4072.

Rokach, L. 2010. Ensemble-Based classifiers. *Artif. Intell. Rev. 33*, 1–39.

Roli, F., Giacinto, G,, and Vernazza, G. 2001. Methods for designing multiple classifier systems. In *Proceedings of the International Workshop on Multiple Classifier Systems.* Lecture Notes in Computer Science, vol. 2096. Springer, 78–87.

Rooney, N. and Patterson, D. 2007. A weighted combination of stacking and dynamic integration. *Pattern Recogn. 40*, 4, 1385–1388.

Rooney, N., Patterson, D., Anand, S., and Tsymbal, A. 2004. Dynamic integration of regression models. In *Proceedings of the International Workshop on Multiple Classifier Systems*. Lecture Notes in Computer Science, vol. 3181. Springer, 164–173.

Rosen, B. E. 1996. Ensemble learning using decorrelated neural networks. *Connect. Sci. 8*, 3–4, 373–383.

RUTA, D. AND GABRYS, B. 2001. Application of the evolutionary algorithms for classifier selection in multiple classifier systems with majority voting. In *Proceedings of the International Workshop on Multiple Classifier Systems.* Lecture Notes in Computer Science, vol. 2096. Springer, 399–408.

SCHAPIRE, R. 1990. The strength of weak learnability. *Mach. Learn. 5*, 2, 197–227.

SCHCLAR, A. AND ROKACH, L. 2009. Random projection ensemble classifiers. In *Proceedings of the International Conference on Enterprise Information Systems (ICEIS'09).* Springer.

SCHCLAR, A., TSINKINOVSKY, A., ROKACH, L., MEISELS, A., AND ANTWARG, L. 2009. Ensemble methods for improving the performance of neighborhood-based collaborative filtering. In *Proceedings of the 3^{rd} ACM Conference on Recommender Systems (RecSys'09).* ACM Press, New York, 261–264.

SHRESTHA, D. L. AND SOLOMATINE, D. P. 2006. Experiments with adaboost.rt, an improved boosting scheme for regression. *Neural Comput. 18*, 1678–1710.

SKALAK, D. B. 1994. Prototype and feature selection by sampling and random mutation hill climbing algorithms. In *Proceedings of the International Conference on Machine Learning*. Morgan Kaufmann, 293–301.

SKOMOROCH, P. 2008. Some datasets available on the web. http://www.datawrangling.com/some-datasets-available-on-the-web

STARK, P. AND PARKER, R. 1995. Bounded-Variable least squares: An algorithm and applications. *Comput. Statist. 10*, 2, 129–141.

STONE, M. 1974. Cross-Validatory choice and assessment of statistical predictions. *J. Roy. Statist. Soc. B36*, 2, 111–147.

STREHL, A. AND GHOSH, J. 2003. Cluster ensembles: A knowledge reuse framework for combining multiple partitions. *J. Mach. Learn. Res. 3*, 583–617.

TAMON, C. AND XIANG, J. 2000. On the boosting pruning problem. In *Proceedings of the European Conference on Machine Learning.* Lecture Notes in Computer Science, vol. 1810. Springer, 404–412.

THOMPSON, S. K. AND SEBER, G. A. 1986. *Adaptive Sampling*. John Wiley & Sons.

TIAN, J., WU, N., CHU, X., AND FAN, Y. 2010. Predicting changes in protein thermostability brought about by single- or multi-site mutations. *BMC Bioinf. 11,* 370.

TODOROVSKI, L. AND DZEROSKI, S. 2003. Combining classifiers with meta decision trees. *Mach. Learn. 50*, 3, 223–249.

TORGO, L. Regression datasets. http://www.liaad.up.pt/ltorgo/Regression/Datasets.html.

TRESP, V. AND TANIGUCHI, M. 1995. Combining estimators using non-constant weighting functions. *Adv. Neural Inf. Process. Syst. 7*, 419–426.

TSANG, I. W., KOCSOR, A., AND KWOK, J. T. 2006. Diversified svm ensembles for large data sets. In *Proceedings of the International Conference on Machine Learning*. Lecture Notes in Artificial Intelligence, vol. 4212. Springer, 792–800.

TSANG, I. W., KWOK, J. T., AND LAI, K. T. 2005. Core vector regression for very large regression problems. In *Proceedings of the International Conference on Machine Learning.* 912–919.

TSOUMAKAS, G., PARTALAS, I., AND VLAHAVAS, I. 2008. A taxonomy and short review of ensemble selection. In *Proceedings of the Workshop on Supervised and Unsupervised Ensemble Methods and Their Applications.*

TSYMBAL, A., PECHENIZKIY, M., AND CUNNINGHAM, P. 2006a. Dynamic integration with random forests. In *Proceedings of the European Conference on Machine Learning (ECML'06).* Lecture Notes in Artificial Intelligence, vol. 4212. Springer, 801–808.

TSYMBAL, A., PECHENIZKIY, M., AND CUNNINGHAM, P. 2006b. Dynamic integration with random forests. Tech. rep. TCD-CS-2006-23, The University of Dublin, Trinity College.

TSYMBAL, A., PECHENIZKIY, M., CUNNINGHAM, P., AND PUURONEN, S. 2008. Dynamic integration of classifiers for handling concept drift. *Inf. Fusion 9*, 1, 56–68.

UEDA, N. AND NAKANO, R. 1996. Generalization error of ensemble estimators. In *Proceedings of the IEEE Conference on Neural Networks*. Vol. 1. 90–95.

VAFAIE, H. AND JONG, K. D. 1993. Robust feature selection algorithms. In *Proceedings of the IEEE Conference on Tools for Artificial Intelligence*. 356–363.

VERIKAS, A., LIPNICKAS, A., MALMQVIST, K., BECAUSKIENE, M., AND GELZINIS, A. 1999. Soft combining of neural classifiers: A comparative study. *Pattern Recogn. Lett. 20*, 4, 429–444.

VLACHOS, P. 2005. Statlib: Datasets archive. http://lib.stat.cmu.edu/datasets/.

WANG, H., FAN, W., YU, P. S., AND HAN, J. 2003. Mining concept-drifting data streams using ensemble classifiers. In *ACM International Conference on Knowledge Discovery and Data Mining.*

WEBB, G. I. AND ZHENG, Z. 2004. Multistrategy ensemble learning: Reducing error by combining ensemble learning techniques. *IEEE Trans. Knowl. Data Engin. 16*, 8, 980–991.

WEYUKER, E. J., OSTRAND, T. J., AND BELL, R. M. 2010. Comparing the effectiveness of several modeling methods for fault prediction. *Empir. Softw. Engin. 15*, 277–295.

WICHARD, J., MERKWIRTH, C., AND OGORZALEK, M. 2003. Building ensembles with heterogeneous models. In *Course of the International School on Neural Nets.*

WILSON, D. R. AND MARTINEZ, T. R. 1997. Improved heterogeneous distance functions. *J. Artif. Intell. Res. 6,* 1–34.

WITTEN, I. H. AND FRANK, E. 2011. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann.

WOLPERT, D. H. 1992. Stacked generalization. *Neural Netw. 5*, 2, 241–259.

WOODS, K. 1997. Combination of multiple classifiers using local accuracy estimates. *IEEE Trans. Pattern Anal. Mach. Intell. 19*, 4, 405–410.

YANG, J. AND HONAVAR, V. 1997. Feature subset selection using a genetic algorithm. *IEEE Trans. Intell. Syst. 13*, 2, 44–49.

YANKOV, D., DECOSTE, D., AND KEOGH, E. 2006. Ensembles of nearest neighbor forecasts. In *Proceedings of the European Conference on Machine Learning*. Lecture Notes in Artificial Intelligence, vol. 4212. Springer, 545–556.

YAO, X., FISCHER, M., AND BROWN, G. 2001. Neural network ensembles and their application to traffic flow prediction in telecommunications networks. *Neural Netw. 1*, 693–698.

YU, Y., ZHOU, Z.-H., AND TING, K. M. 2007. Cocktail ensemble for regression. In *Proceedings of the IEEE International Conference on Data Mining.* 721–726.

ZEMEL, R. S. AND PITASSI, T. 2001. *Advances in NeuralInformation Processing Systems.* Vol. 13. MIT Press, Chapter A gradient-based boosting algorithm for regression problems, 696–702.

ZENOBI, G. AND CUNNINGHAM, P. 2001. 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*. Lecture Notes in Computer Science, vol. 2167. Springer, 576–587.

ZHANG, C.-X., ZHANG, J.-S., AND WANG, G.-W. 2008. An empirical study of using rotation forest to improve regressors. *Appl. Math. Comput. 195*, 2, 618–629.

ZHANG, J., ZOU, Y., AND FAN, Y. 2009. Embedded neural network to model-based permanent magnet synchronous motor diagnostics. In *Proceedings of the Power Electronics and Motion Control Conference*. 1813–1817.

ZHAO, Q.-L., JIANG, Y.-H., AND XU, M. 2009. A fast ensemble pruning algorithm based on pattern mining process. *Data Min. Knowl. Discov. 19*, 277–292.

ZHOU, Z.-H. AND TANG, W. 2003. Selective ensemble of decision trees. In *Proceedings of the International Conference on Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing.* Lecture Notes in Artificial Intelligence, vol. 2639. Springer, 476–483.

ZHOU, Z.-H., WU, J., AND TANG, W. 2002. Ensembling neural networks: Many could be better than all. *Artif. Intell. 137*, 239–263.

Received January 2011; revised June 2011; accepted August 2011