Ensemble Approaches for Regression: a Survey

João M. Moreira [a,*], Carlos Soares [b,c], Alípio M. Jorge [b,c] and Jorge Freire de Sousa [a]

[a] *Faculty of Engineering, University of Porto, Rua Dr. Roberto Frias, s/n 4200-465 Porto PORTUGAL*

[b] *Faculty of Economics, University of Porto, Rua Dr. Roberto Frias, s/n 4200-464 Porto PORTUGAL*

[c] *LIAAD, INESC Porto L.A., R. de Ceuta, 118, 6, 4050-190, Porto PORTUGAL*

Abstract

This paper discusses approaches from different research areas to ensemble regression. The goal of ensemble regression is to combine several models in order to improve the prediction accuracy on learning problems with a numerical target variable. The process of ensemble learning for regression can be divided into three phases: the generation phase, in which a set of candidate models is induced, the pruning phase, to select of a subset of those models and the integration phase, in which the output of the models is combined to generate a prediction. We discuss different approaches to each of these phases, categorizing them in terms of relevant characteristics and relating contributions from different fields. Given that previous surveys have focused on classification, we expect that this work will provide a useful overview of existing work on ensemble regression and enable the identification of interesting lines for further research.

Key words: ensembles, regression, supervised learning

1 Introduction

Ensemble learning typically refers to methods that generate several models which are combined to make a prediction, either in classification or regression

* Tel.: +351225081639; fax: +351225081538
Email address: `jmoreira@fe.up.pt` (João M. Moreira).

problems. This approach has been the object of a significant amount of research in recent years and good results have been reported (e.g., [1–3]). The advantage of ensembles with respect to single models has been reported in terms of increased robustness and accuracy [4].

Most work on ensemble learning focuses on classification problems. However, techniques that are successful for classification are often not directly applicable for regression. Therefore, although both are related, ensemble learning approaches have been developed somehow independently. Therefore, existing surveys on ensemble methods for classification [5,6] are not suitable to provide an overview of existing approaches for regression.

This paper surveys existing approaches to ensemble learning for regression. The relevance of this paper 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, which makes it quite hard for a researcher to be aware of all contributions that are relevant to his/her own work. Therefore, besides 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. Hopefully, this organization will enable the identification of opportunities for further research and facilitate the classification of new approaches.

In the next section, we provide a general discussion of the process of ensemble learning. This discussion will lay out the basis according to which the remaining sections of the paper will be presented: ensemble generation (Sect. 3), ensemble pruning (Sect. 4) and ensemble integration (Sect. 5). Sect. 6 concludes the paper with a summary.

2 Ensemble Learning for Regression

In this section we provide a more accurate definition of ensemble learning and provide terminology. Additionally, we present a general description of the process of ensemble learning and describe a taxonomy of different approaches, both of which define the structure of the rest of the paper. Next we discuss the experimental setup for ensemble learning. Finally, we analyze the error decompositon of ensemble learning methods for regression.

2.1 Definition

First of all we need to define clearly what ensemble learning is, and to define a taxonomy of methods. As far as we know, there is no widely accepted definition of ensemble learning. Some of the existing definitions are partial in the sense that they focus just on the classification problem or on part of the ensemble learning process [7]. For these reasons we propose the following definition:

Definition 1 *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. In the first place, contrary to the informal definition given at the beginning of the paper, this one covers not only ensembles in supervised learning (both classification and regression problems), but also in unsupervised learning, namely the emerging research area of ensembles of clusters [8].

Additionally, it clearly separates ensemble and divide-and-conquer approaches. This last family of approaches split the input space in several sub-regions and train separately each model in each one of the sub-regions. With this approach the initial problem is converted in the resolution of several simpler sub-problems.

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

More formally, an ensemble $\mathcal{F}$ is composed of a set of predictors of a function f denoted as $\hat{f}_i$.

$$\mathcal{F} = \{\hat{f}_i, i = 1, ..., k\}. \tag{1}$$

The resulting ensemble predictor is denoted as $\hat{f}_f$.

2.1.1 The Ensemble Learning Process

The ensemble process can be divided into three steps [9] (Fig. 1), usually referred as the overproduce-and-choose approach. The first step is *ensemble generation*, which consists of generating a set of models. It often happens that, during the first step, a number of redundant models are generated. In the *ensemble pruning* step, the ensemble is pruned by eliminating some of the models generated earlier. 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 for new cases, based on the predictions of the base models.

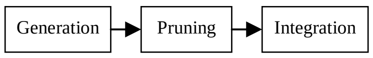

Fig. 1. Ensemble learning model

Our characterization of the ensemble learning process is slightly more detailed than the one presented by Rooney et al. [10]. For those authors, ensemble learning consists on 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 named direct, and can be seen as a particular case of the model presented in Fig. 1, named overproduce-and-choose.

Ensemble pruning has been reported, at least in some cases, to reduce the size of the ensembles obtained without degrading the accuracy. Pruning has also been added to direct methods successfully increasing the accuracy [11,12]. A subject to be discussed further in Sect. 4.

2.1.2 Taxonomy and Terminology

Concerning the categorization of the different approaches to ensemble learning, we will follow mainly the taxonomy presented by the same authors [10]. They divide ensemble generation approaches into *homogeneous*, if all the models were generated using the same induction algorithm, and *heterogeneous*, otherwise.

Ensemble integration methods, are classified by some authors [10,13] 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 from the ensemble the most promising model(s) and the prediction of the ensemble is based on the selected model(s) only. Here, we use, instead, the classification of constant vs. non-constant weighting functions given by Merz [14]. 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. In Table 1 we list several groups of synonyms, extended from a previous list by Kuncheva [5]. The first column contains the most frequently used terms in this paper.

Table 1
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

2.2 Experimental setup

The experimental setups used in ensemble learning methods are very different depending on communities and authors. Our aim is to propose a general framework rather than to do a survey on the different experimental setups described in the literature. The most common approach is to split the data into three parts: (1) the training set, used to obtain the base predictors; (2) the validation set, used for assessment of the generalization error of the base predictors; and (3) the test set, used for assessment of 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. [15] propose to split 50% for training, 25% for validation and the remaining 25% to use as test set. This strategy works for large data sets, let's say, data sets with more than one thousand examples. For large data sets we propose the use of this approach mixed with cross-validation. To do this, and for this particular partition (50%, 25%, and 25%), the data set is randomly divided in four equal parts, two of them being used as training set, another one as validation set and the last one as test set. This process is repeated using all the combinations of training, validation and test sets among the four parts. With this partition there are twelve combinations. For smaller data sets, the percentage of data used for training must be higher. It can be 80%, 10% and 10%. In this case the number of combinations is ninety. The main advantage is to train the base predictors with more examples (it can be critical for small data sets) but it has the disadvantage of increasing the computational cost. The process can be repeated several times in order to obtain different sample values for the evaluation criterion, namely the *mse* (eq. 3).

2.3 Regression

In this paper we assume a typical regression problem. Data consists of a set of n examples of the form $\{(\mathbf{x_1}, f(\mathbf{x_1})), \ldots, (\mathbf{x_n}, f(\mathbf{x_n}))\}$. The goal is to induce a function $\hat{f}$ from the data, where

$$\hat{f} : X \rightarrow \Re, \text{ where, } \hat{f}(\mathbf{x}) = f(\mathbf{x}), \forall \mathbf{x} \in X, \tag{2}$$

where f represents the unknown true function. The algorithm used to obtain the $\hat{f}$ function is called induction algorithm or learner. The $\hat{f}$ function is called model or predictor. The usual goal for regression is to minimize a squared error loss function, namely the mean squared error (mse),

$$mse = \frac{1}{n}\sum_{i}^{n}(\hat{f}(\mathbf{x_i}) - f(\mathbf{x_i}))^2. \tag{3}$$

2.4 Understanding the generalization error of ensembles

To accomplish the task of ensemble generation, it is necessary to know the characteristics that the ensemble should have. Empirically, it is stated by several authors that a good ensemble is the one with accurate predictors and making errors in different parts of the input space. For the regression problem it is possible to decompose the generalization error in different components, which can guide the process to optimize the ensemble generation.

Here, the functions are represented, when appropriate, without the input variables, just for the sake of simplicity. For example, instead of $f(\mathbf{x})$ we use f. We closely follow Brown [16].

Understanding the ensemble generalization error enables us to know which characteristics should the ensemble members have in order to reduce the overall generalization error. The generalization error decomposition for regression is straightforward. What follows is about the decomposition of the *mse* (eq. 3). Despite the fact that the majority of the works were presented in the context of neural network ensembles, the results presented in this section are not dependent of the induction algorithm used.

Geman et al. present the bias/variance decomposition for a single neural network [17]:

$$E\{[\hat{f} - E(f)]^2\} = [E(\hat{f}) - E(f)]^2 + E\{[\hat{f} - E(\hat{f})]^2\}. \tag{4}$$

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:

$$mse(f) = bias(f)^2 + var(f). \tag{5}$$

Krogh & Vedelsby describe the ambiguity decomposition, for an ensemble of k neural networks [18]. Assuming that $\hat{f}_f(\mathbf{x}) = \sum_{i=1}^{k}[\alpha_i \times \hat{f}_i(\mathbf{x})]$ (see Sect. 5.1) where $\sum_{i=1}^{k}(\alpha_i) = 1$ and $\alpha_i \geq 0, i = 1, ..., k$, they show that the error for a single example is:

$$(\hat{f}_f - f)^2 = \sum_{i=1}^{k}[\alpha_i \times (\hat{f}_i - f)^2] - \sum_{i=1}^{k}[\alpha_i \times (\hat{f}_i - \hat{f}_f)^2]. \tag{6}$$

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 non negative. 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 $\mathbf{x}$ (omitted in the formulae just for the sake of simplicity, as previously referred). Two full proofs of the ambiguity decomposition [18] are presented in [16].

Later, Ueda & Nakano presented the bias/variance/covariance decomposition of the generalization error of ensemble estimators [19]. In this decomposition it is assumed that $\hat{f}_f(\mathbf{x}) = \frac{1}{k} \times \sum_{i=1}^{k}[\hat{f}_i(\mathbf{x})]$:

$$E[(\hat{f}_f - f)^2] = \overline{bias}^2 + \frac{1}{k} \times \overline{var} + (1 - \frac{1}{k}) \times \overline{covar}, \tag{7}$$

where

$$\overline{bias} = \frac{1}{k} \times \sum_{i=1}^{k}[E_i(f_i) - f], \tag{8}$$

$$\overline{var} = \frac{1}{k} \times \sum_{i=1}^{k}\{E_i\{[\hat{f}_i - E_i(\hat{f}_i)]^2\}\}, \tag{9}$$

$$\overline{covar} = \frac{1}{k \times (k-1)} \times \sum_{i=1}^{k} \sum_{j=1, j \neq i}^{k} E_{i,j}\{[\hat{f}_i - E_i(\hat{f}_i)][\hat{f}_j - E_j(\hat{f}_j)]\}. \tag{10}$$

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 provides a good discussion on the relation between ambiguity and covariance [16]. An important result obtained from the study of this relation is the confirmation that it is not possible to maximize the ensemble ambiguity without affecting the ensemble bias component as well, i.e., it is not possible to maximize the ambiguity component and minimize the bias component simultaneously.

The discussion of the present section is usually referred in the context of ensemble diversity, i.e., the study on the degree of disagreement between the base predictors. Many of the above statements are related to the well known statistical problem of point estimation. This discussion is also related with the multi-collinearity problem that will be discussed in Sect. 5.

3 Ensemble generation

The goal of ensemble generation is to generate a set of models, $\mathcal{F} = \{\hat{f}_i, i = 1, ..., k\}$. 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 best covered area of ensemble learning in the literature. See, for example, the state of the art surveys from Dietterich [7], or Brown et al. [20]. In this section we mainly follow the former [7]. In homogeneous ensembles, the models are generated using the same algorithm. Thus, as explained in the following sections, diversity can be achieved by manipulating the data (Section 3.1) or by 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 higher diversity [21]. The problem is the lack of control on the diversity of the ensemble 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 the use of the overproduce-and-choose approach. Using this approach the diversity is guaranteed in the pruning phase [22]. Another approach, commonly followed, combines the two approaches, by using different induction algorithms mixed with the use of different parameter sets [23,10] (Sect. 3.2.1). Some authors claim that the use of heterogeneous ensembles improves the performance of homogeneous ensemble generation. Note that heterogeneous ensembles can use homogeneous ensemble models as base learners.

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 targets.

3.1.1 Subsampling from the training set

These methods have in common that the models are obtained using different subsamples from the training set. This approach generally assumes that the algorithm is unstable, i.e., 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 [24,7]. However, some of the methods based on subsampling (e.g., bagging and boosting) have been successfully applied to algorithms usually regarded as stable, such as Support Vector Machines (SVM) [25].

One of the most popular of such methods is *bagging* [26]. 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 sampling uniformly m examples with replacement (some examples appear several times while others do not appear at all). Both Breiman [26] and Domingos [27] give insights on why does bagging work.

Based on [28], Freund & Schapire present the AdaBoost (ADAptive BOOSTing) algorithm, the most popular *boosting algorithm* [29]. The main idea is that it is possible to convert a weak learning algorithm into one that achieves arbitrarily high accuracy. A weak learning algorithm is one that performs slightly better than random prediction. This conversion is done by combining the estimations of several predictors. Like in bagging [26], 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 the examples, but in the following iterations examples with more inaccurate predictions have higher probability of being selected. In each new iteration there are more 'difficult examples' in the training set. Despite boosting has been originally developed for classification, several algorithms have been proposed for regression but none has emerged as being the appropriate one [30].

Parmanto et al. describe the *cross-validated committees* technique for neural networks ensemble generation using υ-fold cross validation [31]. The main idea is to use as ensemble the models obtained by the use of the υ training sets on the cross validation process.

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 j is generated by replacing the original representation $\{(\mathbf{x_i}, f(\mathbf{x_i}))$ into a new one $\{(\mathbf{x'_i}, f(\mathbf{x_i}))$. There are two types of approaches. The first one is feature selection, i.e., $\mathbf{x'_i} \subset \mathbf{x_i}$. In the second approach, the representation is obtained by applying some transformation to the original attributes, i.e., $\mathbf{x'_i} = g(\mathbf{x_i})$.

A simple feature selection approach is the *random subspace* method, consisting of a random selection [32]. The models in the ensemble are independently constructed using a randomly selected feature subset. Originally, decision trees were used as base learners and the ensemble was called *decision forests* [32]. The final prediction is the combination of the predictions of all the trees in the forest.

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 from a random feature selection [33]. The author uses neural networks for the classification problem. He reports better results using this approach than using the popular bagging and AdaBoost methods. In [34] the search method is a wrapper like hill-climbing strategy. The criteria used to select the feature subsets are the minimization of the individual error and the maximization of ambiguity (Sect. 2.4).

A feature selection approach can also be used to generate ensembles for algorithms that are stable with respect to the training set but unstable w.r.t. the set of features, namely the nearest neighbors induction algorithm. In [35] the feature subset selection is done using adaptive sampling in order to reduce the risk of discarding discriminating information. Compared to random feature selection, this approach reduces diversity between base predictors but increases their accuracy.

A simple transformation approach is *input smearing* [36]. 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:

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

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. In this work just the numeric input variables are smeared even if the nominal ones could also be smeared using a different strategy. Results compare favorably to bagging. A similar approach called BEN - *Bootstrap Ensemble with Noise*, was previously presented by Raviv & Intrator [37].

Rodriguez et al. [3] 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 higher 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 assures diversity, (decision trees are sensitive to the rotation of the axis) and accuracy (PCA concentrates in a few features most of the information contained in the data). The authors claim that rotation forests outperform bagging, AdaBoost and random forests (to be discussed further away in Sect. 3.2.2). However, the adaptation of rotation forests for regression does not seem to be straightforward.

3.1.3 Manipulating the output targets

The manipulation of the output targets can also be used to generate different training sets. However, not much research follows this approach and most of it focus on classification.

An exception is the work of Breiman, called *output smearing* [38]. The basic idea is to add Gaussian noise to the target variable of the training set, in the same way as it is done for input features in the input smearing method (Sect. 3.1.2). Using this approach it is possible to generate as many models as desired. Although it was originally proposed using CART trees as base models, it can be used with other base algorithms. The comparison between output smearing and bagging shows a consistent generalization error reduction, even if not outstanding.

An alternative approach consists of the following steps. First it generates a model using the original data. Second, it generates a model that estimates the error of the predictions of the first model and generates an ensemble that combines the prediction of the previous model with the correction of the current one. Finally, it iteratively generates models that predict the error of the current ensemble and then 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 was called *iterated bagging* [39]. Iterated bagging reduces generalization error when compared with bagging, mainly due to the bias reduction during the iteration process.

3.2 Model generation manipulation

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

3.2.1 Manipulating the parameter sets

Each induction algorithm is sensitive to the values of the input parameters. The degree of sensitivity of the induction algorithm is different for different input parameters. To maximize the diversity of the models generated, one should focus on the parameters which the algorithm is most sensitive to.

Neural network ensemble approaches quite often use different initial weights to obtain different models. This is done because the resulting models vary significantly with different initial weights [40]. Several authors, like Rosen, for example, use randomly generated seeds (initial weights) to obtain different models [41], while other authors mix this strategy with the use of different number of layers and hidden units [42,43].

The k-nearest neighbors ensemble proposed by Yankov et al. [44] has just two members. They differ on the number of nearest neighbors used. They are both sub-optimal. 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 Sect. 5.2.1).

3.2.2 Manipulating the induction algorithm

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

Rosen [41] generates ensembles of neural networks by sequentially training networks, adding a decorrelation penalty to the error function, to increase diversity. Using this approach, the training of each network tries to minimize a function that has a covariance component, thus decreasing the generalization error of the ensemble, as stated in [19]. This was the first approach using the decomposition of the generalization error made by Ueda & Nakano [19] (Sect. 2.4) to guide the ensemble generation process. Another sequential method to generate ensembles of neural networks is called SECA (*Stepwise Ensemble Construction Algorithm*) [30]. It uses bagging to obtain the training set for each neural network. The neural networks are trained sequentially. The process stops when adding another neural network to the current ensemble increases the generalization error.

The *Cooperative Neural Network Ensembles* (CNNE) method [45] also uses a sequential approach. In this work, the ensemble begins with two neural networks and then, iteratively, CNNE tries to minimize the ensemble error firstly by training the existing networks, then by adding a hidden node to an existing neural network, and finally by adding a new neural network. Like in Rosen's approach, the error function includes a term representing the correlation between the models in the ensemble. Therefore, to maximize the diversity, all the models already generated are trained again at each iteration of the process. The authors test their method not only on classification datasets but also on one regression data set, with promising results.

Tsang et al. [46] propose an adaptation of the CVM (Core Vector Machines) algorithm [47] that maximizes 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 by the CVM algorithm. This approach can be related to AdaBoost because higher weights are given to instances which are incorrectly classified in previous iterations.

Note that the sequential approaches mentioned above add a penalty term to the error function of the learning algorithm. This sort of added penalty has been also used in the parallel method *Ensemble Learning via Negative Correlation* (ELNC) to generate neural networks that are learned simultaneously so that the overall quality of the ensemble is taken into account [48].

Parallel approaches that exchange information during the process typically integrate the learning algorithm with an evolutionary framework. Opitz & Shavlik [49] present the ADDEMUP (*Accurate anD Diverse Ensemble-Maker giving United Predictions*) method to generate ensembles of neural networks. In this approach, the fitness metric for each network weights the accuracy of the network and the diversity of this network within the ensemble. The bias/variance decomposition presented by Krogh & Vedelsby [18] is used. Genetic operators of mutation and crossover are used to generate new models from previous ones. The new networks are trained emphasizing misclassified examples. 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. A similar approach is the *Evolutionary Ensembles with Negative Correlation Learning* (EENCL) method, which combines the ELNC method with an evolutionary programming framework [1]. In this case, the only genetic operator used is mutation, which randomly changes the weights of an existing neural network. The EENCL has two advantages in common with other parallel approaches. First, the models are trained simultaneously, emphasizing specialization and cooperation among individuals. Second the neural network ensemble generation is done according to the integration method used, i.e., the learning models and the ensemble integration are part of the same process, allowing possible interactions between them. Additionally, the ensemble size is obtained automatically in the EENCL method.

A parallel approach in which each learning process does not take into account the quality of the others but in which there is exchange of information about the models is given by the *cooperative coevolution of artificial neural network ensembles* method [4]. 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 sub-populations of models evolve independently; and (2) by the use of a multiobjective evaluation fitness measure, combining network and ensemble fitness. Multiobjective is a quite well known research area in the operational research community. The authors use a multiobjective algorithm based on the concept of Pareto optimality. Other groups of objectives (measures) besides the cooperation ones are: objectives of performance, regularization, diversity and ensemble objectives. The authors do 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 authors just studied the classification one. 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 & Li formulate an *infinite ensemble* based on the SVM (Support Vector Machines) algorithm [50]. The main idea is to create a kernel that embodies all the possible models in the hypothesis space. The SVM algorithm is then used to generate a linear combination of all 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 [2] uses an algorithm for induction of decision trees which 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 (Sect. 3.1). The strength of the method is the combined use of boostrap sampling and random feature selection.

3.2.3 Manipulating the model

Given a learning process that produces one single model M, it can potentially be transformed into an ensemble approach by producing a set of models M_i from the original model M. Jorge & Azevedo have proposed a post-bagging approach for classification [51] 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 by a large committee of classifiers constructed as described above. Experimental results on 12 datasets show a consistent, although slight, advantage over the singleton learning process. The same authors also propose an approach with some similarities to boosting [52]. Here, the rules in the original model M are iteratively reassessed, filtered and reordered according to their performance on the training set. Again, experimental results show minor but consistent improvement over using the original model, and also show a reduction on 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. Model manipulation has only been applied in the realm of classification association rules, a highly modular representation. Applying to other kinds of models, such as decision trees or neural networks, does not seem trivial. It could be, however, easily tried with regression rules.

3.3 A discussion on ensemble generation

Two relevant issues arise from the discussion above. 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.

In general, existing results indicate that ensemble methods are competitive when compared to individual models. For instance, random forests are consistently among the best three models in the benchmark study by Meyer et al. [53], which included many different algorithms.

However, there is little knowledge about the strengths and weaknesses of each method, given that the results reported in different papers are not comparable because of the use of different experimental setups [45,4].

It is possible to distinguish the most interesting/promising methods for some of the most commonly used induction algorithms. For decision trees, bagging [26] by its consistency and simplicity, and random forest [2] by its accuracy, are the most appealing ensemble methods. Despite obtaining good results on classification problems, the rotation forests method [3] has not been adapted for regression yet.

For neural networks, methods based on negative correlation are particularly appealing, due to their theoretical foundations [16] and good empirical results. EENCL is certainly an influent and well studied method on neural network ensembles [1]. Islam et al. [45] and Garcia-Pedrajas et al. [4] also present interesting methods.

One important line of work is the adaptation of the methods described here to other algorithms, namely support vector regression and k-nearest neighbors. Although some attempts have been made, there is still much work to be done.

Additionally, we note that most research focuses on one specific approach to build the ensemble (e.g., subsampling from the training set or manipulating the induction algorithm). Further investigation is necessary on the gains that can be achieved by combining several approaches.

4 Ensemble pruning

Ensemble pruning consists of eliminating models from the ensemble, with the aim of improving its predictive ability or reducing costs. In the overproduce and choose approach it is the choice step. In the direct approach, ensemble pruning, is also used to reduce computational costs and, if possible, to increase prediction accuracy [11,54]. Bakker & Heskes claim that clustering models (later described in Sect. 4.5) summarizes the information on the ensembles, thus giving new insights on the data [54]. Ensemble pruning can also be used to avoid the multi-collinearity problem [42,43] (to be discussed in Sect. 5).

The ensemble pruning process has many common aspects with feature selection, namely, the search algorithms that can be used. In this section, the ensemble pruning methods are classified and presented according to the used search algorithm: exponential, randomized and sequential; plus the ranked pruning and the clustering algorithms. It finishes with a discussion on ensemble pruning, where experiments comparing some of the algorithms described along the paper are presented.

4.1 Exponential pruning algorithms

When selecting a subset of k models from a pool of K models, the searching space has $2^K - 1$ non-empty subsets. The search of the optimal subset is a NP-complete problem [55]. According to Martínez-Muñoz & Suárez it becomes intractable for values of $K > 30$ [12]. Perrone & Cooper suggest this approach for small values of K [42].

Aksela presents seven pruning algorithms for classification [56]. One of them can also be used for regression. It calculates the correlation of the errors for each pair of predictors in the pool and then it selects the subset with minimal mean pairwise correlation. This method implies the calculus of the referred metric for each possible subset.

4.2 Randomized pruning algorithms

Partridge & Yates describe the use of a genetic algorithm for ensemble pruning but with poor results [57].

Zhou et al. state that it can be better to use just part of the models from an ensemble than to use all of them [11]. Their work on neural network ensembles, called GASEN (Genetic Algorithm based Selective ENsemble) starts by the assignment of a random weight to each one of the base models. Then it employs a genetic algorithm to evolve those weights in order to characterize the contribution of the corresponding model to the ensemble. Finally it selects the networks whose weights are bigger than a predefined threshold. Empirical results on ten regression problems show that GASEN outperforms bagging and boosting both in terms of bias and variance. Results on classification are not so promising. Following this work, Zhou & Tang successfully applied GASEN to build ensembles of decision trees [58].

Ruta & Gabrys use three randomized algorithms to search for the best subset of models [59]: genetic algorithms, tabu search and population-based incremental learning. The main result of the experiments on three classification data sets, using a pool of $K = 15$, was that the three algorithms obtained most of best selectors when compared against exhaustive search. These results may have been conditioned by the small size of the pool.

4.3 Sequential pruning algorithms

The sequential pruning algorithms iteratively change one solution by adding or removing models. Three types of search algorithms are used:

- Forward: if the search begins with an empty ensemble and adds models to the ensemble in each iteration;
- Backward: if the search begins with all the models in the ensemble and eliminates models from the ensemble in each iteration;
- Forward-backward: if the selection can have both forward and backward steps.

4.3.1 Forward selection

Forward selection starts with an empty ensemble and iteratively adds models with the aim of decreasing the expected prediction error.

Coelho & Von Zuben describe two forward selection algorithms called Cw/oE - constructive without exploration, and CwE - constructive with exploration [60]. However, to use a more conventional categorization, the algorithms will be renamed Forward Sequential Selection with Ranking (FSSwR) and Forward Sequential Selection (FSS), respectively. The FSSwR ranks all the candidates with respect to its performance on a validation set. Then, it selects the candidate at the top until the performance of the ensemble decreases. In the FSS algorithm, each time a new candidate is added to the ensemble, all candidates are tested and it is selected the one that leads to the maximal improvement of the ensemble performance. When no model in the pool improves the ensemble performance, the selection stops. This approach is also used in [9]. These algorithms were firstly described for ensemble pruning by Perrone & Cooper [42].

Partridge & Yates present another forward selection algorithm similar to the FSS [57]. The main difference is that the criterion for the inclusion of a new model is a diversity measure. The model with higher diversity than the ones already selected is also included in the ensemble. The ensemble size is an input parameter of the algorithm.

Another similar approach is presented in [61]. At each iteration it tests all the models not yet selected, and selects the one that reduces most the ensemble generalization error on the training set. Experiments to reduce ensembles generated using bagging are promising even if overfitting could be expected since the minimization of the generalization error is done on the training set.

4.3.2 Backward selection

Backward selection starts with all the models in the ensemble and iteratively removes models with the aim of decreasing the expected prediction error.

Coelho & Von Zuben describe two backward selection algorithms called Pw/oE - pruning without exploration, and PwE - pruning with exploration [60]. Like for the forward selection methods, they will be renamed Backward Sequential Selection with Ranking (BSSwR) and Backward Sequential Selection (BSS), respectively. In the first one, the candidates are previously ranked according to their performance in a validation set (like in FSSwR). The worst is removed. If the ensemble performance improves, the selection process continues. Otherwise, it stops. BSS is related to FSS in the same way BSSwR is related to FSSwR, i.e., it works like FSS but using backward selection instead of forward selection.

4.3.3 Mixed forward-backward selection

In the forward and backward algorithms described by Coelho & Von Zuben, namely the FSSwR, FSS, BSSwR and BSS, the stopping criterion assumes that the evaluation function is monotonic [60]. However, in practice, this cannot be guaranteed. The use of mixed forward and backward steps aims to avoid the situations where the fast improvement at the initial iterations does not allow to explore solutions with slower initial improvements but with better final results.

Moreira et al. describe an algorithm that begins by randomly selecting a pre-defined number of k models [62]. At each iteration one forward step and one backward step are given. The forward step is equivalent to the process used by FSS, i.e., it selects the model from the pool that most improves the accuracy of the ensemble. At this step, the ensemble has $k+1$ models. The second step selects the k models with higher ensemble accuracy, i.e, 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 & Dietterich present an algorithm called reduce-error pruning with back fitting [63]. This algorithm is similar to the FSS in the two first iterations. After the second iteration, i.e., when adding the third candidate and the following ones, a back fitting step is given. Consider C_1, C_2 and C_3 as the included candidates. Firstly it removes C_1 from the ensemble and tests the addition of each of the remaining candidates $C_i (i > 3)$ to the ensemble. It repeats this step for C_2 and C_3. It chooses the best of the tested sets. Then it executes further iterations until a pre-defined number of iterations is reached.

4.4 Ranked pruning algorithms

The ranked pruning algorithms sort the models according to a certain criterion and generate an ensemble containing the top k models in the ranking. The value of k is either given or determined on the basis of a given criterion, namely, a threshold, a minimum, a maximum, etc.

Partridge & Yates rank the models according to the accuracy [57]. Then, the k most accurate models are selected. As expected, results are not good because there is no guarantee of diversity. Kotsiantis & Pintelas use a similar approach [64]. For each model a t-test is done for comparison of the accuracy with the most accurate model. Tests are carried out using randomly selected 20% of the training set. If the p-value of the t-test is lower than 5%, the model is rejected. The use of heterogeneous ensembles is the only guarantee of diversity. Rooney et al. use a metric that tries to balance accuracy and diversity [10].

Perrone & Cooper describe an algorithm that removes similar models from the pool [42]. It uses the correlation matrix of the predictions and a pre-defined threshold to identify them.

4.5 Clustering algorithms

The main idea of clustering is to group the models in several clusters and choose representative models (one or more) from each cluster.

Lazarevic uses the prediction vectors made by all the models in the pool [65]. The k-means clustering algorithm is used over these vectors to obtain clusters of similar models. Then, for each cluster, the algorithms are ranked according to their accuracy and, beginning by the least accurate, the models are removed (unless their disagreement with the remaining ones overcomes a pre specified threshold) until the ensemble accuracy on the validation set starts decreasing. The number of clusters (k) is an input parameter of this approach, i.e., in practice this value must be tested by running the algorithm for different k values or, like in Lazarevic's case, an algorithm is used to obtain a default k [65]. The experimental results reported are not conclusive.

Coelho & Von Zuben [60] use the ARIA - Adaptive Radius Immune Algorithm, for clustering. This algorithm does not require a pre specified k parameter. Just the most accurate model from each cluster is selected.

Partridge & Yates compare three of the approaches previously described [57]: (1) Ranked according to the accuracy; (2) FSS using a diversity measure; and (3) a genetic algorithm. The results are not conclusive because just one data set is used. The FSS using a diversity measure gives the best result. However, as pointed out by the authors, the genetic algorithm result, even if not very promising, can not be interpreted as being less adapted for ensemble pruning. The result can be explained by the particular choices used for this experiment. Ranked according to the accuracy gives the worst result, as expected.

Roli et al. compare several pruning algorithms using one data set with three different pools of models [9]. In one case, the ensemble is homogeneous (they use 15 neural networks trained using different parameter sets), in the other two cases they use heterogeneous ensembles. The algorithms tested are: FSS selecting the best model in the first iteration and selecting randomly a model for the first iteration, BSS, tabu search, Giacinto & Roli's clustering algorithm [66], and some others. The tabu search and the FSS selecting the best model in the first iteration give good results for the three different pools of models.

Coelho & Von Zuben also use just one data set to compare FSSwR, FSS, BSSwR, BSS and the clustering algorithm using ARIA [60]. Each one of these algorithms are tested with different integration approaches. Results for each one of the tested ensemble pruning algorithms give similar results, but for different integration methods. Ensembles obtained using the clustering algorithm and BSS have higher diversity.

The ordered bagging algorithm by Martínez-Muñoz & Suárez is compared with FSS using, also, just one data set [12]. The main advantage of ordered bagging is the meaningfully lower computational cost. The differences in accuracy are not meaningful.

Ruta & Gabrys compare a genetic algorithm, a population-based incremental learning algorithm and tabu search on three classification data sets [59]. Globally, differences are not meaningful between the three approaches. The authors used a pool of fifteen models, not allowing to explore the differences between the three methods.

All of these benchmark studies discussed are for ensemble classification. It seems that more sophisticated algorithms like the tabu search, genetic algorithms, population based incremental learning, FSS, BSS or clustering algorithms are able to give better results, as expected. All of them use a very small number of data sets, limiting the generalization of the results.

5 Ensemble integration

Now that we have described the process of ensemble generation, we move to the next step: how to combine the strengths of the models in one ensemble to obtain one single answer, i.e., ensemble integration.

For regression problems, ensemble integration is done using a linear combination of the predictions. This can be stated as

$$\hat{f}_f(\mathbf{x}) = \sum_{i=1}^{k}[h_i(\mathbf{x}) * \hat{f}_i(\mathbf{x})], \tag{12}$$

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

Merz divides the integration approaches in constant and non-constant weighting functions [14]. In the first case, the $h_i(\mathbf{x})$ are constants (in this case α_i will be used instead of $h_i(\mathbf{x})$ in eq. 12); while in the second one, the weights vary according to the input values $\mathbf{x}$.

When combining predictions, a possible problem is the existence of multi-collinearity between the predictions of the ensemble models. As a consequence of the multi-collinearity problem, the confidence intervals for the α_i coefficients will be wide, i.e., the estimators of the coefficients will have high variance [14]. This happens because to obtain the α_i's we must determine the inverse of a linearly dependent matrix.

A common approach is to handle multi-collinearity in the ensemble generation (Sect. 3) or in the ensemble pruning (Sect. 4) phases. If the principles referred in Sect. 2.4, namely the accuracy and diversity ones are assured, then it is possible, if not to avoid completely, at least ameliorate this problem.

This section follows Merz classification. It finishes with a discussion on ensemble integration methods.

5.1 Constant weighting functions

Constant weighting integration functions always use the same set of coefficients, independently of the input for prediction. They are summarized in Fig. 2. Some methods use the test data to obtain the α_i weights of the integration function (eq. 12).

Next, we describe the main constant weighting functions following closely Merz [14].

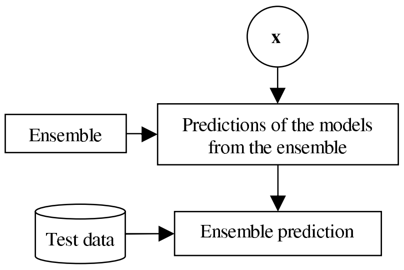

Fig. 2. Constant weighting functions model

The BEM - Basic Ensemble Method [42] uses as estimator for the target function

$$\hat{f}_{BEM}(\mathbf{x}) = \sum_{i=1}^{k}[\frac{1}{k} * \hat{f}_i(\mathbf{x})]. \tag{13}$$

This formula can be written as

$$\hat{f}_{BEM}(\mathbf{x}) = f(\mathbf{x}) - \frac{1}{k}\sum_{i=1}^{k} m_i(\mathbf{x}), \tag{14}$$

where

$$m_i(\mathbf{x}) = f(\mathbf{x}) - \hat{f}_i(\mathbf{x}). \tag{15}$$

BEM assumes that the $m_i(\mathbf{x})$ are mutually independent with zero mean.

To address this issue, Perrone & Cooper propose the GEM - Generalized Ensemble Method [42]. For GEM, the estimator is

$$\hat{f}_{GEM}(\mathbf{x}) = \sum_{i=1}^{k}[\alpha_i * \hat{f}_i(\mathbf{x})] = f(\mathbf{x}) + \sum_{i=1}^{k}[\alpha_i * m_i(\mathbf{x})], \tag{16}$$

where

$$\begin{aligned} \sum_{j=1}^{k} \alpha_i &= 1, \\ \alpha_i &= \frac{\sum_{j=1}^{k} C_{ij}^{-1}}{\sum_{l=1}^{k}\sum_{j=1}^{k} C_{lj}^{-1}}, \\ C_{ij} &= E[m_i(\mathbf{x}) * m_j(\mathbf{x})]. \end{aligned}$$

The drawback of this method is the multi-collinearity problem, since it is necessary to calculate the inverse matrix C^{-1}. The multi-collinearity problem can be avoided by pruning the ensemble [42] (Sect. 4).

The well known linear regression (LR) model is another possible combination method. The predictor is the same as in 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 $E[\hat{f}_i(\mathbf{x})] \simeq E[f(\mathbf{x})]$ [67]. It would be necessary if predictors were meaningfully biased.

All the methods discussed so far suffer from the multi-collinearity problem with the exception of BEM. Next we discuss methods that avoid this problem.

Caruana et al. embed the ensemble integration phase in the ensemble selection one [22]. By selecting with replacement the models from the pool to include in the ensemble, and using the simple average as the weighting function, 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).

Breiman presents the stacked regression [68] method based on the well known stacked generalization framework [69] firstly presented for the classification problem. Given a $\mathcal{L}$ learning set with M examples, the goal is to obtain the α_i coefficients that minimize

$$\sum_{j=1}^{M}[f(x_j) - \sum_{i=1}^{k} \alpha_i * \hat{f}_i(\mathbf{x}_j)]^2. \tag{17}$$

To do so, the learning set used to obtain the α_i coefficients will be the same one used to train the $\hat{f}_i$ estimators, over-fitting the data. This problem is solved by using υ-fold cross-validation,

$$\sum_{j=1}^{M}[f(x_j) - \sum_{i=1}^{k} \alpha_i * \hat{f}_i^{(-\upsilon)}(\mathbf{x}_j)]^2. \tag{18}$$

The second problem is the possible existence of multi-collinearity between the $\hat{f}_i$ predictors. Breiman presents several approaches to obtain the α_i coefficients but concludes that a method that gives consistently good results is the minimization of the above equation under the constraints $\alpha_i \geq 0, i = 1, ..., k$ [68]. One of the methods tried by Breiman to obtain the α_i coefficients is ridge regression, a regression technique for solving badly conditioned problems. but results were not promising [68]. An important result of Breiman is the empirical observation that, in most of the cases, many of the α_i weights are zero.

This result supports the use of ensemble pruning as a second step after the ensemble generation (Sect. 4).

Merz & Pazzani use principal component regression to avoid the multi-collinearity problem [70]. The PCR* method obtains the principal components (PC) and, then, it selects the number of PCs to use. Once the PCs are ordered as a function of the variation they can explain, the search of the number of PCs to use is much simplified. The choice of the correct number of PCs is important to avoid under-fitting or over-fitting.

Evolutionary algorithms have also been used to obtain the α_i coefficients [71]. The used approach is globally better than BEM and GEM on twenty five classification data sets. Compared to BEM wins nineteen, draws one and looses five. Compared to GEM wins seventeen, draws one and looses seven. These results were obtained by direct comparison, i.e., without statistical validity.

The main study comparing constant weighting functions is presented by Merz [14]. 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 by Kivinen & Warmuth [72]), ridge regression, constrained regression (Merz [14] uses the bounded variable least squares method from [73]), stacked constrained regression (with ten partitions) and PCR*. Two of the three experiments reported by Merz are summarized next. The first experiment used an ensemble of twelve models on eight regression data sets: six of the models were generated using MARS [74] and the other six using the neural network back-propagation algorithm. The three globally best functions were CR, EG and PCR*. The second experiment tests how the functions perform with many correlated models. The author uses neural network ensembles of size ten and fifty. Just three data sets were used. The PCR* function presents more robust results. See [14] to get details on the experiments and their results.

5.2 Non-constant weighting functions

The non-constant weighting functions use different coefficients according to the input for prediction. They can be static (defined at learning time) or dynamic (defined at prediction time). The static ones can use two different approaches: (1) to assign models to predefined regions, this is the divide-and-conquer approach already discussed in Sect. 3.1.1; or (2) to define the areas of expertise for each model, also called static selection [13], i. e., for each predictor from the ensemble it is defined the input subspace where the predictor is expert. In the dynamic approach the $h_i(\mathbf{x})$ weights from eq. 12 are obtained on the fly based on the performances of the base predictors on data similar to $\mathbf{x}$ obtained from the training set.

5.2.1 The approach by areas of expertise

The work on meta decision trees [75] for classification induces meta decision trees using as variables meta-attributes that are previously calculated for each example. The target variable of the meta tree is the classifier to recommend. In practice, the meta decision tree, instead of predicting, recommends a predictor. Despite this work has been developed for classification, it could be easily adapted for regression by an appropriate choice of the meta attributes.

Yankov et al. use support vector machines with the gaussian kernel to select the predictor to use from an ensemble with two models [44].

5.2.2 The dynamic approach

In the dynamic approach, the predictor(s) selection is done on the fly, i.e., given the input vector for the prediction task, it chooses the expected best predictor(s) to accomplish this task. While in the approach by areas of expertise these areas are previously defined, in the dynamic approach the areas are defined on the fly. Selection can be seen as a kind of pruning on the fly. The most usual way to select the predictor(s) is by evaluating their performances on similar data from the training set, for a chosen performance metric. Often the similar data is obtained by the use of the k-nearest neighbors with the Euclidean distance [76]. Puuronen et al. [77] use the more advanced weighted k-nearest neighbor.

Rooney et al. [10] adapt for regression the dynamic integration methods originally presented by Puuronen et al. [77] for classification. Dynamic selection (DS) selects the predictor with less cumulative error on the k-nearest neighbors set. Dynamic weighting (DW) assigns a weight to each base model according to its localized performance on the k-nearest neighbors set [10] and the final prediction is based on the weighted average of the predictions of the related models. Dynamic weighting with selection (DWS) is similar to DW but the predictors with cumulative error in the upper half of the error interval are discarded. From the three methods tested the DWS one using just the subset of the most accurate predictors for the final prediction gets the best results.

Wang et al. use weights $h_i(\mathbf{x})$ (see eq. 12) inversely proportional to $\hat{f}_i(\mathbf{x})$ expected error [78]. This approach is similar to the variance based weighting presented in [79].

Figure 3 summarizes the dynamic approach. Given an input vector $\mathbf{x}$, firstly it selects similar data. Then, according to the performance of the models on this similar data, a number k_1 of models are selected from the ensemble $\mathcal{F}$ (Eq. 1. Merz describes in detail this approach, namely the use of a performance matrix to evaluate locally the models [23]. It consists of a $m \times k$ matrix, where m is the number of past examples and k is the number of models in $\mathcal{F}$. Each cell has a performance measure of each model for each example obtained from the models' past predictions and the respective real target values for these examples. In regression, this measure can be, for instance, the squared error, the absolute error, or other performance measure. If $k_1 = 1$ then $\hat{f}_f$ is the identity function. It is the case of the DS method. If $k_1 > 1$ (like in the DW method, for example) then the integration method uses the performances of similar data (sd) obtained from the test data (td) used to estimate the $h_i(\mathbf{x})$ weights. Despite the good results reported with this approach, Didaci et al. show, for the case of the selection of just one model for prediction, that they are still far away from the best possible (the oracle), i.e., they are still far away from the use of the model with the best accuracy prediction for each example. The tests were done on five classification data sets [80].

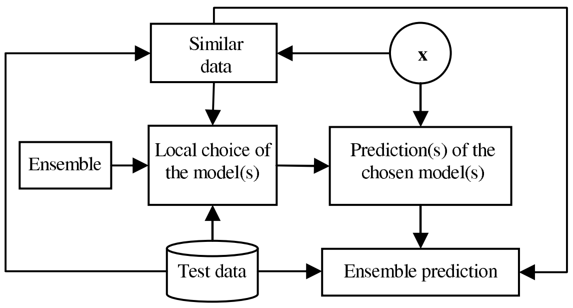
Similar data
x
Ensemble
Local choice of the model(s)
Prediction(s) of the chosen model(s)
Test data
Ensemble prediction

Fig. 3. Dynamic approach model

5.3 A discussion on ensemble integration

The works on how to handle the multi-collinearity problem, quite common in the nineties, became less frequent in the last years, apparently because most of the effort shifted to the generation phase. The approach seems to have changed from "what integration function to use for a given ensemble?" to "how to generate the ensemble for a given integration function?". In all the works highlighted in Sect. 3.3 constant weighting functions were used. It is already known by the decomposition of the generalization error (Sect. 2.4) which are the expected characteristics of the ensemble when using constant weighting functions. The question now is: "how to generate ensembles for non-constant weighting functions?".

The experiments described by Merz (Sect. 2) were published in 1998 [14]. However, since 1995, maybe due to the advances in the studies on the generalization ensemble error (Sect. 2.4), the ensemble generation research was driven towards the quality of the ensembles [41,1,11,3]. These examples show that an important part of the problems at the integration phase can be solved by a joint design of the generation, pruning (when appropriate) and the integration phases.

The main disadvantage of constant weighting functions is that the α_i weights, being equal for all the input space, can, at least theoretically, be less adequate for some parts of the input space. This is the main argument for using non-constant weighting functions [81]. This argument can be particularly true for time changing phenomena [78].

Ensemble integration approaches can also be classified as selection or combination ones [13]. In the selection approach the final prediction is obtained by using just one predictor, while in the combination one the final prediction is obtained by combining predictions of two or more models. Kuncheva presents an hybrid approach between the selection and the combination ones [13]. It uses paired t-hypothesis test to verify if there is one predictor meaningfully better than the others. If positive, it uses the best predictor, if not it uses a combination approach.

An approach that can be explored and seems to be promising is to combine different ensemble integration methods. The method wMetaComb [82] uses a weighted average to combine stacked regression (described in Sect. 5.1) and the DWS dynamic method (Sect. 5.2.2). The weights are determined based on the error performance of both methods (see [82] for details). Tests on 30 regression data sets never looses against stacked regression (it wins 5 and draws the remaining 25) and looses 3 against DWS (it wins 13 and draws 14).

6 Conclusions

The use of ensemble methods has as main advantages the increase in accuracy and robustness, when compared to the use of a single model. This makes ensemble methods particularly suited for applications where small improvements of the predictions have important impact.

For ensemble learning, as for other research areas, methods for regression and for classification have different solutions, at least partially. This paper is focused in the regression problem, less referred in the literature. The methods for ensemble learning have, typically, three phases: generation, pruning (not always) and integration.

The generation phase aims to obtain an ensemble of models. It can be classified as homogeneous or as heterogeneous. This classification depends on the induction algorithms used. If just one is used, the ensemble is classified as

Table 2
Main homogeneous ensemble generation methods

Method	**Reference**	**Algorithm**	**Class/Regr**
Bagging	[26]	Unst. learners	yes / yes
AdaBoost	[29]	Unst. learners	yes / ?
Random forests	[2]	Decis. trees	yes / yes
Rotation forests	[3]	Decis. trees	yes / ?
EENCL	[1]	ANN	yes / yes
CNNE	[45]	ANN	yes / yes
Coop. Coev.	[4]	ANN	yes / ?

homogeneous, otherwise it is classified as heterogeneous. The most successful methods for ensemble generation are developed for unstable learners, i.e., learners that are sensitive to changes in the training set, namely decision trees or neural networks. Table 2 summarizes some of the most important methods on homogeneous ensemble generation. The mark ? means that there is not, until the moment, promising versions of this algorithm for regression (the case of AdaBoost), or that these methods are not even adapted and tested for regression (rotation forests and cooperative co-evolution), and consequently it is not known how these methods could work for regression.

Ensemble pruning aims to select from a pool of models a subset in order 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 from a set, a subset in order to optimize a given objective function. Like in the feature subset selection case, randomized heuristics, such as evolutionary algorithms or tabu search, seem to be very effective.

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 non-constant weighting functions. As it was previously underlined, constant weighting functions are the most used ones. Maybe because it is easier to generate ensembles in order to minimize known generalization error functions. Since non-constant 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 paper described the complete process for ensemble based regression. As it is shown, at each step there are many challenging problems to be solved, and many ideas still need theoretical and experimental development. We believe that this survey provides a thorough road map that can serve as a stepping stone to new ideas for research.

Acknowledgements

This work was partially supported by FCT (POCT/TRA/61001/2004) and FEDER e Programa de Financiamento Plurianual de Unidades de I&D.

References

[1] Yong Liu, Xin Yao, and Tetsuya Higuchi, “Evolutionary ensembles with negative correlation learning”, *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 4, pp. 380–387, 2000.

[2] Leo Breiman, “Random forests”, *Machine Learning*, vol. 45, pp. 5–32, 2001.

[3] Juan J. Rodríguez, Ludmila I. Kuncheva, and Carlos J. Alonso, “Rotation forest: a new classifier ensemble”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 10, pp. 1619–1630, 2006.

[4] Nicolás García-Pedrajas, César Hervás-Martínez, and Domingo Ortiz-Boyer, “Cooperative coevolution of artificial neural network ensembles for pattern classification”, *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 3, pp. 271–302, 2005.

[5] Ludmila I. Kuncheva, *Combining pattern classifiers*, Wiley, 2004.

[6] Romesh Ranawana and Vasile Palade, “Multi-classifier systems: review and a roadmap for developers”, *International Journal of Hybrid Intelligent Systems*, vol. 3, no. 1, pp. 35–61, 2006.

[7] Thomas G. Dietterich, “Machine-learning research: four current directions”, *AI magazine*, vol. 18, no. 4, pp. 97–136, 1997.

[8] Alexander Strehl and Joydeep Ghosh, “Cluster ensembles - a knowledge reuse framework for combining multiple partitions”, *Journal of Machine Learning Research*, vol. 3, pp. 583–617, 2003.

[9] Fabio Roli, Giorgio Giacinto, and Gianni Vernazza, “Methods for designing multiple classifier systems”, in *International Workshop on Multiple Classifier Systems*. 2001, vol. LNCS 2096, pp. 78–87, Springer.

[10] Niall Rooney, David Patterson, Sarab Anand, and Alexey Tsymbal, “Dynamic integration of regression models”, in *International Workshop on Multiple Classifier Systems*. 2004, vol. LNCS 3181, pp. 164–173, Springer.

[11] Zhi-Hua Zhou, Jianxin Wu, and Wei Tang, “Ensembling neural networks: many could be better than all”, *Artificial Intelligence*, vol. 137, pp. 239–263, 2002.

[12] Gonzalo Martínez-Muñoz and Alberto Suárez, "Pruning in ordered bagging ensembles", in *International Conference on Machine Learning*, 2006, pp. 609–616.

[13] Ludmila I. Kuncheva, "Switching between selection and fusion in combining classifiers: an experiment", *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, vol. 32, no. 2, pp. 146–156, 2002.

[14] Cristopher J. Merz, *Classification and regression by combining models*, Phd thesis, University of California - U.S.A., 1998.

[15] Trevor Hastie, Robert Tibshirani, and Jerome H. Friedman, *The elements of statistical learning: data mining, inference, and prediction*, Springer series in statistics. Springer, 2001.

[16] Gavin Brown, *Diversity in neural network ensembles*, Phd thesis, University of Birmingham - United Kingdom, 2004.

[17] Stuart Geman, Elie Bienenstock, and Rene Doursat, "Neural networks and the bias/variance dilemma", *Neural Computation*, vol. 4, no. 1, pp. 1–58, 1992.

[18] Anders Krogh and Jesper Vedelsby, "Neural network ensembles, cross validation, and active learning", *Advances in Neural Information Processing Systems*, vol. 7, pp. 231–238, 1995.

[19] Naonori Ueda and Ryohei Nakano, "Generalization error of ensemble estimators", in *IEEE International Conference on Neural Networks*, 1996, vol. 1, pp. 90–95.

[20] Gavin Brown, Jeremy L. Wyatt, Rachel Harris, and Xin Yao, "Diversity creation methods: a survey and categorisation", *Information Fusion*, vol. 6, pp. 5–20, 2005.

[21] Geoffrey I. Webb and Zijian Zheng, "Multistrategy ensemble learning: reducing error by combining ensemble learning techniques", *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 8, pp. 980–991, 2004.

[22] Rich Caruana, Alexandru Niculescu-Mozil, Geoff Crew, and Alex Ksikes, "Ensemble selection from libraries of models", in *International Conference on Machine Learning*, 2004.

[23] Cristopher J. Merz, "Dynamical selection of learning algorithms", in *International Workshop on Artificial Intelligence and Statistics*, D. Fisher and H.-J. Lenz, Eds. 1996, vol. Learning from Data: Artificial Intelligence and Statistics V, Springer-Verlag.

[24] Leo Breiman, "Heuristics of instability and stabilization in model selection", *Annals of Statistics*, vol. 24, no. 6, pp. 2350–2383, 1996.

[25] Hyun-Chul Kim, Shaoning Pang, Hong-Mo Je, Daijin Kim, and Sung-Yang Bang, "Constructing support vector machine ensemble", *Pattern Recognition*, vol. 36, no. 12, pp. 2757–2767, 2003.

[26] Leo Breiman, “Bagging predictors”, *Machine Learning*, vol. 26, pp. 123–140, 1996.

[27] Pedro Domingos, “Why does bagging work? a bayesian account and its implications”, in *International Conference on Knowledge Discovery and Data Mining*. 1997, pp. 155–158, AAAI Press.

[28] R. Schapire, “The strength of weak learnability”, *Machine learning*, vol. 5, no. 2, pp. 197–227, 1990.

[29] Y. Freund and R. Schapire, “Experiments with a new boosting algorithm”, in *International Conference on Machine Learning*, 1996, pp. 148–156.

[30] P.M. Granitto, P.F. Verdes, and H.A. Ceccatto, “Neural network ensembles: evaluation of aggregation algorithms”, *Artificial Intelligence*, vol. 163, no. 2, pp. 139–162, 2005.

[31] Bambang Parmanto, Paul W. Munro, and Howard R. Doyle, “Reducing variance of committee prediction with resampling techniques”, *Connection Science*, vol. 8, no. 3, 4, pp. 405–425, 1996.

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

[33] David W. Opitz, “Feature selection for ensembles”, in *16th National Conference on Artificial Intelligence*, Orlando - U.S.A., 1999, pp. 379–384, AAAI Press.

[34] Gabriele Zenobi and Pádraig Cunningham, “Using diversity in preparing ensembles of classifiers based on different feature subsets to minimize generalization error”, in *European Conference on Machine Learning*. 2001, vol. LNCS 2167, pp. 576–587, Springer.

[35] Carlotta Domeniconi and Bojun Yan, “Nearest neighbor ensemble”, in *International Conference on Pattern Recognition*, 2004, vol. 1, pp. 228–231.

[36] Eibe Frank and Bernhard Pfahringer, “Improving on bagging with input smearing”, in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. 2006, pp. 97–106, Springer.

[37] Yuval Raviv and Nathan Intrator, “Bootstrapping with noise: an effective regularization technique”, *Connection Science*, vol. 8, no. 3, 4, pp. 355–372, 1996.

[38] Leo Breiman, “Randomizing outputs to increase prediction accuracy”, *Machine Learning*, vol. 40, no. 3, pp. 229–242, 2000.

[39] Leo Breiman, “Using iterated bagging to debias regressions”, *Machine Learning*, vol. 45, no. 3, pp. 261–277, 2001.

[40] John F. Kolen and Jordan B. Pollack, “Back propagation is sensitive to initial conditions”, Tech. Report TR 90-JK-BPSIC, The Ohio State University, 1990.

[41] Bruce E. Rosen, "Ensemble learning using decorrelated neural networks", *Connection Science*, vol. 8, no. 3, 4, pp. 373–383, 1996.

[42] Michael P. Perrone and Leon N. Cooper, "When networks disagree: ensemble methods for hybrid neural networks", in *Neural Networks for Speech and Image Processing*, R.J. Mammone, Ed. Chapman-Hall, 1993.

[43] Sherif Hashem, *Optimal linear combinations of neural networks*, Phd thesis, Purdue University, 1993.

[44] Dragomir Yankov, Dennis DeCoste, and Eamonn Keogh, "Ensembles of nearest neighbor forecasts", in *European Conference on Machine Learning*. 2006, vol. LNAI 4212, pp. 545–556, Springer.

[45] Md. Monirul Islam, Xin Yao, and Kazuyuki Murase, "A constructive algorithm for training cooperative neural network ensembles", *IEEE Transactions on Neural Networks*, vol. 14, no. 4, pp. 820–834, 2003.

[46] Ivor W. Tsang, Andras Kocsor, and James T. Kwok, "Diversified svm ensembles for large data sets", in *European Conference on Machine Learning*. 2006, vol. LNAI 4212, pp. 792–800, Springer.

[47] Ivor W. Tsang, James T. Kwok, and Kimo T. Lai, "Core vector regression for very large regression problems", in *International Conference on Machine Learning*, 2005, pp. 912–919.

[48] Y. Liu and X. Yao, "Ensemble learning via negative correlation", *Neural Networks*, vol. 12, pp. 1399–1404, 1999.

[49] David W. Opitz and Jude W. Shavlik, "Generating accurate and diverse members of a neural-network ensemble", *Advances in Neural Information Processing Systems*, vol. 8, pp. 535–541, 1996.

[50] Hsuan-Tien Lin and Ling Li, "Infinite ensemble learning with support vector machines", in *European Conference on Machine Learning*. 2005, vol. LNAI 3720, pp. 242–254, Springer.

[51] Alípio M. Jorge and Paulo J. Azevedo, "An experiment with association rules and classification: post-bagging and conviction", in *Discovery science*, Singapore, 2005, vol. LNCS 3735, pp. 137–149, Springer.

[52] Paulo J. Azevedo and Alípio Mário Jorge, "Iterative reordering of rules for building ensembles without relearning", in *10th International Conference on Dicovery Science*. 2007, vol. LNCS 4755, pp. 56–67, Springer.

[53] David Meyer, Friedrich Leisch, and Kurt Hornik, "The support vector machine under test", *Neurocomputing*, vol. 55, no. 1-2, pp. 169–186, 2003.

[54] Bart Bakker and Tom Heskes, "Clustering ensembles of neural network models", *Neural Networks*, vol. 16, no. 2, pp. 261–269, 2003.

[55] Christino Tamon and Jie Xiang, "On the boosting pruning problem", in *European Conference on Machine Learning*. 2000, vol. LNCS 1810, pp. 404–412, Springer.

[56] Matti Aksela, "Comparison of classifier selection methods for improving committee performance", in *International Workshop on Multiple Classifier Systems*. 2003, vol. LNCS 2709, pp. 84–93, Springer.

[57] D. Partridge and W. B. Yates, "Engineering multiversion neural-ney systems", *Neural Computation*, vol. 8, no. 4, pp. 869–893, 1996.

[58] Zhi-Hua Zhou and Wei Tang, "Selective ensemble of decision trees", in *International Conference on Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing*. 2003, vol. LNAI 2639, pp. 476–483, Springer.

[59] Dymitr Ruta and Bogdan Gabrys, "Application of the evolutionary algorithms for classifier selection in multiple classifier systems with majority voting", in *International Workshop on Multiple Classifier Systems*. 2001, vol. LNCS 2096, pp. 399–408, Springer.

[60] Guilherme P. Coelho and Fernando J. Von Zuben, "The influence of the pool of candidates on the performance of selection and combination techniques in ensembles", in *International Joint Conference on Neural Networks*, 2006, pp. 10588–10595.

[61] Daniel Hernández-Lobato, Gonzalo Martínez-Muñoz, and Alberto Suárez, "Pruning in ordered regression bagging ensembles".

[62] João M. Moreira, Jorge Freire Sousa, Alípio M. Jorge, and Carlos Soares, "An ensemble regression approach for bus trip time prediction", in *Meeting of the EURO Working Group on Transportation*, 2006, pp. 317–321.

[63] Dragos D. Margineantu and Thomas G. Dietterich, "Pruning adaptive boosting", in *International Conference on Machine Learning*, 1997, pp. 211–218.

[64] S.B. Kotsiantis and P.E. Pintelas, "Selective averaging of regression models", *Annals of Mathematics, Computing & Teleinformatics*, vol. 1, no. 3, pp. 65–74, 2005.

[65] Aleksandar Lazarevic, "Effective pruning of neural network classifier ensembles", in *International Joint Conference on Neural Networks*, 2001, pp. 796–801.

[66] Giorgio Giacinto and Fabio Roli, "Design of effective neural network ensembles for image classification purposes", *Image and Vision Computing*, vol. 19, no. 9, pp. 699–707, 2001.

[67] Michael LeBlanc and Robert Tibshirani, "Combining estimates in regression and classification", *Journal of the American Statistical Association*, vol. 91, pp. 1641–1650, 1996.

[68] Leo Breiman, "Stacked regressions", *Machine Learning*, vol. 24, pp. 49–64, 1996.

[69] David H. Wolpert, "Stacked generalization", *Neural Networks*, vol. 5, no. 2, pp. 241–259, 1992.

[70] Cristopher J. Merz and Michael J. Pazzani, "A principal components approach to combining regression estimates", *Machine Learning*, vol. 36, pp. 9–32, 1999.

[71] Domingo Ortiz-Boyer, César Hervás-Martínez, and Nicolás García-Pedrajas, "Cixl2: A crossover operator for evolutionary algorithms based on population features", *Journal of Artificial Intelligence Research*, vol. 24, pp. 1–48, 2005.

[72] Jyrki Kivinen and Manfred K. Warmuth, "Exponentiated gradient versus gradient descent for linear predictors", *Information and Computation*, vol. 132, no. 1, pp. 1–63, 1997.

[73] P. Stark and R. Parker, "Bounded-variable least squares: an algorithm and applications", *Computational Statistics*, vol. 10, no. 2, pp. 129–141, 1995.

[74] J.H. Friedman, "Multivariate adaptive regression splines", *The Annals of Statistics*, vol. 19, no. 1, pp. 1–141, 1991.

[75] Ljupco Todorovski and Saso Dzeroski, "Combining classifiers with meta decision trees", *Machine Learning*, vol. 50, no. 3, pp. 223–249, 2003.

[76] Kevin Woods, "Combination of multiple classifiers using local accuracy estimates", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 4, pp. 405–410, 1997.

[77] Seppo Puuronen, Vagan Terziyan, and Alexey Tsymbal, "A dynamic integration algorithm for an ensemble of classifiers", in *International Symposium on Methodologies for Intelligent Systems*. 1999, vol. LNCS 1609, pp. 592–600, Springer.

[78] Haixun Wang, Wei Fan, Philip S. Yu, and Jiawei Han, "Mining concept-drifting data streams using ensemble classifiers", in *ACM International Conference on Knowledge Discovery and Data Mining*, 2003.

[79] Volker Tresp and Michiaki Taniguchi, "Combining estimators using non-constant weighting functions", *Advances in Neural Information Processing Systems*, vol. 7, pp. 419–426, 1995.

[80] Luca Didaci, Giorgio Giacinto, Fabio Roli, and Gian Luca Marcialis, "A study on the performances of dynamic classifier selection based on local accuracy estimation", *Pattern Recognition*, vol. 38, no. 11, pp. 2188–2191, 2005.

[81] Antanas Verikas, Arunas Lipnickas, Kerstin Malmqvist, Marija Becauskiene, and Adas Gelzinis, "Soft combining of neural classifiers: a comparative study", *Pattern Recognition Letters*, vol. 20, no. 4, pp. 429–444, 1999.

[82] Niall Rooney and David Patterson, "A weighted combination of stacking and dynamic integration", *Pattern Recognition*, vol. 40, no. 4, pp. 1385–1388, 2007.