Pruning in Ordered Regression Bagging Ensembles

Daniel Hernández-Lobato, Gonzalo Martínez-Muñoz and Alberto Suárez

***Abstract*— An efficient procedure for pruning regression ensembles is introduced. Starting from a bagging ensemble, pruning proceeds by ordering the regressors in the original ensemble and then selecting a subset for aggregation. Ensembles of increasing size are built by including first the regressors that perform best when aggregated. This strategy gives an approximate solution to the problem of extracting from the original ensemble the minimum error subensemble, which we prove to be NP-hard. Experiments show that pruned ensembles with only 20% of the initial regressors achieve better generalization accuracies than the complete bagging ensembles. The performance of pruned ensembles is analyzed by means of the bias-variance decomposition of the error.**

I. Introduction

The combination of the outputs produced by an ensemble of regressors has been shown to be a consistent way to improve accuracy in many regression problems. Ensemble techniques make use of instabilities in the algorithms that generate the base learners to construct a set of diverse regressors whose combined action can improve the generalization performance of a single unit [1], [2], [3], [4].

One of the most widely used methods to construct regression ensembles is bagging [1] (*bootstrap sampling and aggregation*). In bagging diversity is achieved by training each regressor with a different bootstrap sample of the original training data [5]. The bootstrap sample has the same size of the training data and is obtained by sampling with replacement from it. On average each bootstrap sample contains 63.2%. of the original data and 36.8% of repeated examples. The final prediction of the ensemble is the average of the responses of its individual members. In bagging, the generalization error usually decreases as more regressors are incorporated into the ensemble. The error approaches asymptotically a constant level at larger ensemble sizes. The explanation for this decrease is the lower error variance achieved by the ensemble as a result of the aggregation process. In many regression problems of interest this asymptotic error is lower than the error of a single regressor constructed with the original training data.

It is often assumed that this asymptotic error is the best performance bagging can produce, and that the complete bagging ensemble should be retained to achieve the highest regression accuracy. However, some investigations show that it is possible to select subensembles that outperform the complete ensemble [6]. This is a reflection of the fact that some of the regressors generated in bagging have a detrimental effect on the regression accuracy and should actually be removed from the ensemble. However, the identification of these regressors is a difficult task. Exploratory experiments show that the performance of a regressor in an ensemble is not based solely on individual properties of the learner. As a matter of fact, the complementariness of the responses of regressors that are combined is a key factor in the effectiveness of the ensemble. This is in agreement with observations made in the literature on the beneficial effects of a controlled amount of diversity and/or negative correlations between regressors in the ensemble [7], [8]. Thus, the problem of selecting the optimal subensemble involves searching in the space of $2^M - 1$ non empty subsets of the original ensemble of size M. This problem can be shown to be NP-hard and its exact solution by exhaustive exploration is unfeasible for typical bagging ensembles.

In this work we introduce a greedy algorithm to address the problem of identifying the optimal subset of regressor from the original bagging ensemble. The algorithm designed reaches an approximate solution in polynomial time. Experiments show that, despite the fact that this solution need not be globally optimal, it is in general a near-optimal local minimum. The strategy used is to modify the order of the aggregation process. In standard bagging, regressors are aggregated in the order in which they are generated from the different bootstrap samples of the original training data. The aggregation order is determined by the bootstrap sampling process and is therefore random. In ordered bagging, the aggregation is delayed until all regressors are generated. Subensembles of increasing size are constructed by incorporating at each iteration the regressor that reduces the training error of the subensemble the most. At some point in the aggregation the process is halted and the resulting subensemble is returned as the final solution. This early stopping in aggregation allows the selection of a subensemble that is smaller and that can improve the generalization performance of the original ensemble. A difficulty, inherent to all machine learning algorithms, is that the optimal subensemble selected on the basis of the training set may have a suboptimal generalization performance. Nonetheless, the experiments carried out show that the optimal subensembles selected on the basis of the training set by the pruning algorithm proposed have good generalization properties in the regression problems investigated. An analysis of their performance can be made in terms of the dependence of the bias and variance contributions to the error on the size of the subensembles.

Owing to the fact that they are composed of less regres-

This work was supported in part by the Spanish *Dirección General de Investigación*, project TIN2004-07676-C02-02. Daniel Hernández-Lobato acknowledges support from the *Comunidad Autónoma de Madrid* under an F.P.I. grant.

Daniel Hernández-Lobato, Gonzalo Martínez-Muñoz and Alberto Suárez are with the Computer Science Department, Escuela Politécnica Superior, Universidad Autónoma de Madrid, C/ Francisco Tomás y Valiente, 11, Madrid 28049 Spain. Emails: {daniel.hernandez, gonzalo.matrinez and alberto.suarez} @uam.es.

sors, these pruned ensembles have lower storage requirements, and shorter response times than the corresponding unpruned ensembles. Efficiency in the use of computational resources is a major issue in current applications of automatic learning, especially in online applications that handle large amounts of data and/or require a fast response.

The article is organized as follows: In Section II, we give a brief review of previous work on regression ensembles that is related to the current investigation. Section III introduces the problem of optimal subensemble selection. In Section IV an approximate near-optimal and efficient solution to this problem based on altering the aggregation order in the bagging ensemble is proposed. A bias-variance analysis of the dependence of the regression error on the size in both unordered and ordered subensembles is also presented. Finally, in order to assess the accuracy of the proposed ensemble pruning method we carry out experiments over a wide variety of regression tasks, including synthetic datasets and problems from real-world applications.

II. Related Work

As a result of the remarkable success of ensemble methods, much research has been devoted to improving the performance of ensembles and to reducing their rather large computational requirements, both in memory and in processing time. In this section we give a short review of ensemble pruning techniques related to the present work.

In [6], Zhou *et al.* propose the use of a genetic algorithm to determine an optimal set of weights for the regressors in the ensemble by minimizing a function that estimates the generalization error of the ensemble. The optimization problem is solved by using a standard Genetic Algorithm with a floating-point scheme for real-valued weights in neural networks ensembles. Then, those nets whose optimized weights are below a specified level are eliminated from the ensemble. The experiments made employed rather small ensembles of 20 regressors.

Another related proposal is made in Ref. [7], where the ensemble is built by simultaneously training a collection of networks using a correlation penalty term in the error function. This penalty term is included to encourage specialization and cooperation among the individual networks. It leads to the generation of regressors that are negatively correlated. Small ensembles with about 10 neural networks can be constructed using this method.

The work presented in Ref. [9] describes how collinearity among the members of the ensemble can have harmful effects on the estimation of the optimal weights for the linear combination that is the output of the ensemble. Two algorithms are proposed to improve the performance of the ensemble by dropping some of the collinear regressors. The first one considers collinearity between the outputs of the regressors and the second one collinearity between their errors. The ensembles built are also rather small (6 regressors).

A different approach to ensemble pruning is to cluster regressors in the ensemble according to their outputs and then to select a single representative member for every cluster that has been identified [10]. The clustered representation does not necessarily improve the performance of the complete ensemble but is more amenable to qualitative analysis and can yield novel insights into the data.

The pruning approach proposed in our work can be used in combination with any parallel ensemble method, where there is no intrinsic order in the aggregation process. Although the regressors are trained without additional terms in the error function, the ordering procedure selects first subensembles of complementary regressors that tend to be negatively correlated as in [7]. The original bagging ensembles considered in our investigation are generally larger than those considered in [7], [9], [6]. However, the pruned ensembles based on ordered bagging, besides being smaller, systematically outperform the original bagging ensembles in the regression problems investigated.

III. Selection of optimal subensembles

Consider a regression problem, where the goal is to learn a predictor of the dependent variable $\mathbf{y} \in \mathbb{R}^q$ as a function of the attributes $\mathbf{x} \in \mathbb{R}^p$ using the training data $Z_{tr} = \{(\mathbf{x}_1, \mathbf{y}_1), ..., (\mathbf{x}_N, \mathbf{y}_N)\}$ that is drawn from a probability distribution $\mathcal{P}(Z)$. In this work we assume $q = 1$, but all the expressions can be easily generalized to the case where $q \geq 1$. Bagging works by aggregating many diverse regressors, each one built using the same learning algorithm from a different bootstrap sample of the original training data. Assume $\hat{f}_i(\mathbf{x}|Z_i)$ is the prediction given by the ith regressor built with Z_i, the ith bootstrap sample of the training data Z_{tr}. The prediction of the ensemble is the average of the individual responses of the M regressors in the ensemble

$$\hat{f}_{ens}^{(M)}(\mathbf{x}) = M^{-1} \sum_{i=1}^{M} \hat{f}_i(\mathbf{x}|Z_i), \; i = 1, 2, \ldots, M. \quad (1)$$

The error of the bagging ensemble is

$$E = \int \left(M^{-1} \sum_{i=1}^{M} \hat{f}_i(\mathbf{x}|Z_i) - f(\mathbf{x}) \right)^2 p(\mathbf{x}) d\mathbf{x}, \quad (2)$$

where $f(\mathbf{x})$ is the target function to approximate, and $p(\mathbf{x})$ is the probability density distribution in attribute space. After some algebra [6], Eq.(2) can be expressed as

$$E = M^{-2} \sum_{i=1}^{M} \sum_{j=1}^{M} C_{ij} \quad (3)$$

where

$$C_{ij} = \int \left(\hat{f}_i(\mathbf{x}|Z_i) - f(\mathbf{x}) \right) \left(\hat{f}_j(\mathbf{x}|Z_j) - f(\mathbf{x}) \right) p(\mathbf{x}) d\mathbf{x}$$

and C_{ii} is the average squared error of the *ith* ensemble member.

Assume that a bagging ensemble composed of M different regressors has been constructed. To prune the regression bagging ensemble we try to select the subensemble composed of u regressors $\{s_1, s_2, \ldots, s_u\}$ that minimizes the error

$$E^{(u)} = u^{-2} \sum_{i=1}^{u} \sum_{j=1}^{u} C_{s_i s_j}. \tag{4}$$

Since the true error is not available in the learning problem, we make the selection of the optimal subensemble based on the training error. The expression for the training error is identical to Eq. (4), where the average over $p(\mathbf{x})$ in the calculation of C_{ij} is replaced by an average over the training data

$$C_{ij}^{(tr)} = \frac{1}{N} \sum_{n=1}^{N} (\hat{f}_i(\mathbf{x}_n|Z_i) - f(\mathbf{x}_n))(\hat{f}_j(\mathbf{x}_n|Z_j) - f(\mathbf{x}_n)). \tag{5}$$

All the information needed for the optimization problem is the matrix C_{ij} estimated over the training set. We expect the estimate $C_{ij}^{(tr)}$ to be similar to the C_{ij} matrix calculated over the true distribution of the data. In doing this we make the assumption that minimizing training error leads to the minimization of generalization error. This is not necessarily the case in actual regression problems. In fact minimizing the training error usually leads to overfitting to the training data, and to the selection of an ensemble whose generalization performance is suboptimal. Indeed, the experiments carried out show that the size of the subensembles that minimize the error on the training data tend to be smaller than the optimal subensembles when the error is estimated on a test set that is independent of the training set. Even assuming that a subensemble with lower generalization error than the original pool of regressors exists, the process of finding the one with the lowest error is complex and implies generating all the possible $2^N - 1$ non empty subensembles of the original ensemble. In the Appendix we show that selecting the subset of regressors in an ensemble that minimizes the mean square error estimated on some data set is an NP-hard problem. Therefore the selection of the optimal subensemble is not generally feasible in practice.

IV. Ordered Bagging

As seen in the previous section, the problem of selecting the optimal subensemble cannot be solved in polynomial time unless NP = P. This leads to the rather disappointing conclusion that, as the number of regressors in the ensemble grows, the complexity of the problem becomes unmanageable. Despite this observation, it may still be possible to find *near-optimal* solutions to the subensemble selection problem in practice.

A. Ordering Algorithm

We follow the classic theory on efficient approximate solutions of NP-complete problems [11] and design a polynomial time greedy algorithm that constructs at each step the best local solution. The algorithm starts with an empty ensemble and then selects at each iteration the regressor that, when incorporated, reduces the training error of the new ensemble the most. The process is very similar to the one used in [12], [13] to order classification ensembles. The regressor selected in the uth iteration is the one that minimizes the expression

$$s_u = \underset{k}{arg\,min}\ u^{-2}(\sum_{i=1}^{u-1}\sum_{j=1}^{u-1} C_{s_i s_j} + 2\sum_{i=1}^{u-1} C_{s_i k} + C_{kk}) \tag{6}$$

where $k \in \{1,..,N\}\backslash\{s_1, s_2..., s_{u-1}\}$ and where $\{s_1, s_2..., s_{u-1}\}$ are the indexes of the regressors that have already been incorporated into the pruned ensemble at iteration $u-1$. Algorithm 1 shows the pseudocode of the proposed algorithm.

Algorithm 1 Ordering Algorithm.

```
Input: Vector of regressors R and training data Z.
1 $M \leftarrow |R|$; $N \leftarrow |Z|$
2 for (i from 1 to $M$)
3     for (j from 1 to $M$)
4         $C_{ij} \leftarrow N^{-1} \sum_{n=1}^{N} \left[ \left( \hat{f}_i(\mathbf{x}_n) - f(\mathbf{x}_n) \right) \left( \hat{f}_j(\mathbf{x}_n) - f(\mathbf{x}_n) \right) \right]$
5 $\mathbf{s} \leftarrow empty\,vector$
6 for (u from 1 to $M$)
7     $minimum \leftarrow +\infty$
8     for ($k$ in $\{1,..,M\}\backslash\{s_1,..,s_{u-1}\}$)
9         $value \leftarrow u^{-2}(\sum_{i=1}^{u-1}\sum_{j=1}^{u-1} C_{s_i s_j} +$
              $2\sum_{i=1}^{u-1} C_{s_i k} + C_{kk})))$
10        if ($value < minimum$) {
11            $s_u \leftarrow k$
12            $minimum \leftarrow value$
13        }
14 return $\mathbf{s}$
Output: An ordered vector of the regressor indexes.
```

Therefore, subensembles of increasing size are built, each with one more element than its predecessor. Because the subensemble generated at iteration *u* includes all the regressors of the subensemble generated at iteration $u-1$, this process can be seen as an ordering of the regressors of the complete ensemble, where each subensemble of size u with $1 \le u \le M$ is built by taking the first u regressors in the ordered sequence. This choice need not be the global optimum, because the optimal subensemble of size *u* (the one with the lowest mean square error over the training data) might not include the regressors of the optimal ensemble of size *u* - 1. However, on typical cases, we expect it to be a near-optimal approximation.

The time-complexity of this algorithm as a function of the number of regressors in the bagging ensemble can be easily estimated. As stated earlier, it is necessary to estimate the matrix C_{ij} on the training set before the ordering process can begin. This preprocessing has a cost $O(M^2 \cdot N)$, where N is the size of training samples and M is the number of regressors in the original ensemble. Then at each one of the M iterations we have to extract the regressor that minimizes expression (6) from the remaining set of regressors. This task has a cost $O(((M+1)-u)\cdot u)$, where $1 \leq u \leq M$ is the current iteration. Therefore the final cost of the greedy algorithm is the sum of each one of the previous costs, resulting in a complexity $O(M^3 + M^2 \cdot N)$.

B. Bias-variance analysis.

In this section we perform a bias-variance analysis to investigate the dependence of the error on the subensemble size. Since the the regressors that make up a bagging ensemble are generated from independent bootstrap samples of the original training data, they can be seen as independent realizations of a random variable drawn from a distribution $\mathcal{P}(\hat{f}_i)$. Taking the expectation over this distribution, and after some algebra, the mean square error of a regression ensemble of size *u* can be expressed as a function of the average bias, the average variance and the average covariance of the individual regressors in the ensemble [14]

$$E(u) = u^{-1}\overline{Var} + (1-u^{-1})\overline{Cov} + \overline{Bias}^2,$$

with the definitions

$$\overline{Bias} = u^{-1}\sum_{i=1}^{u} Bias(\hat{f}_i), \qquad \overline{Var} = u^{-1}\sum_{i=1}^{u} Var(\hat{f}_i),$$
$$\overline{Cov} = (u-1)^{-1}u^{-1}\sum_{j\neq i} Cov(\hat{f}_i,\hat{f}_j).$$

In standard (randomly ordered) bagging the expected variances and biases of all regressors are equal. Also, because the regressors are built with independent bootstrap samples of the training data their covariances are zero. Therefore, the regression error of a randomly ordered bagging ensemble of size *u* is simply

$$E = u^{-1}Var + Bias^2 \tag{7}$$

where Var and $Bias$ are the expected variance and bias of a regressor drawn from the distribution $\mathcal{P}(\hat{f}_i)$.

The ordering procedure has the effect of altering the distribution of the regressors that are part of the ensemble at iteration *u*. Instead of being independent of the subensemble size this distribution changes as new regressors from the ordered sequence are aggregated into the ensemble. In what follows $\mathcal{P}(\hat{f}_i, u)$ denotes the distribution of the regressors that are part of the ordered ensemble at iteration *u*. Carrying out the bias-variance error decomposition for the ordered bagging we have that the error for an ordered subensemble of size *u* is

$$E = u^{-1}Var(u) + Bias(u)^2 \tag{8}$$

where $Var(u)$ and $Bias(u)$ are the expected variance and bias of a regressor drawn from the distribution $\mathcal{P}(\hat{f}_i, u)$. As a result of the ordering strategy we expect the values of $Bias(u)$ and $Var(u)$ (the average bias and variance of a regressor in an ordered subensemble of size u) to be lower than $Bias$ and Var (the average bias and variance bias and variance of a regressor in a bagging ensemble), for intermediate subensemble sizes. In this manner ordered bagging achieves lower errors than standard bagging for subensembles of size $u = 1, 2, \ldots, (M-1)$. Notice that when $u = M$, where M is the size of the original ensemble, $\mathcal{P}(\hat{f}_i, u) = \mathcal{P}(\hat{f}_i)$, and the errors of ordered and standard bagging ensembles are equal.

The lowest error that standard bagging achieves is $lim_{M\to\infty}E = Bias^2 \geq 0$. Hence, it is possible for the subensemble at iteration *u* to have a lower error than this asymptotic limit if the inequality

$$u^{-1}Var(u) + Bias(u)^2 < Bias^2 \tag{9}$$

is satisfied. This equality obtains if the algorithm selects a set of regressors from the complete ensemble with a low bias and variance. In the experiments carried out it is seen that the inequality is fulfilled for sufficiently large subensembles.

Figure 1 shows the error curves for an ensemble composed of 100 neural networks for the regression problem *Boston Housing* [15]. These curves display the dependence of the mean square error in regression on the number of regressors in the ensemble. The four different curves correspond to train and test errors in both randomly ordered and ordered bagging ensembles. Each curve is the result of averaging the 10-fold cross-validation error estimates for the regression problem, with 10 different partitions of the data. The features displayed by the error curves are representative of all the regression problems investigated.

As anticipated, when the aggregation order in bagging is random the error decreases monotonically as more regressors are added to the ensemble. The decrease slows down for larger ensembles and asymptotically approaches a saturation level. If the aggregation order is modified according to the procedure described, the error curves show an initial decrease, which is initially steeper than in the randomly ordered ensemble. At intermediate ensemble sizes the error curves display a fairly broad minimum. After this minimum the error increases slowly and eventually approaches the error level of the complete bagging ensemble from below. The improvements in regression accuracy in the ordered ensemble that are observed in the training error curve are also observed in the error curve estimated on the test set. In particular, the minimum in the test error curve in an ensemble whose aggregation order is chosen according to the training data is significatively lower than the best result of randomly ordered bagging. However, it is also apparent in Figure 1 the minimum appears earlier in the aggregation process (i.e. for smaller ensembles) in the training error curve than in the error curve for the test set. This means that it is difficult to exactly estimate from the training data where the minimum in the error curve for the test data lies. Nevertheless, the error curves are rather flat around the minimum, so that overestimating or underestimating the size of the ensemble that corresponds to the minimum does not have a large effect on the generalization performance.

Figures 2 and 3 show how the variance and bias contributions to the error of the ensemble members vary as a function of the number of regressors in the ensemble for both train and test datasets in the *Boston Housing* regression problem.

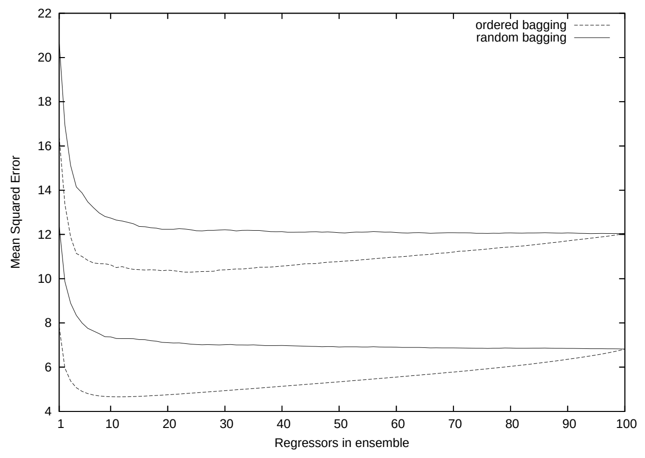

Fig. 1. Average train and test mean square error for Boston data set, for ordered bagging and randomly ordered bagging as a function of the number of regressors in the ensemble.

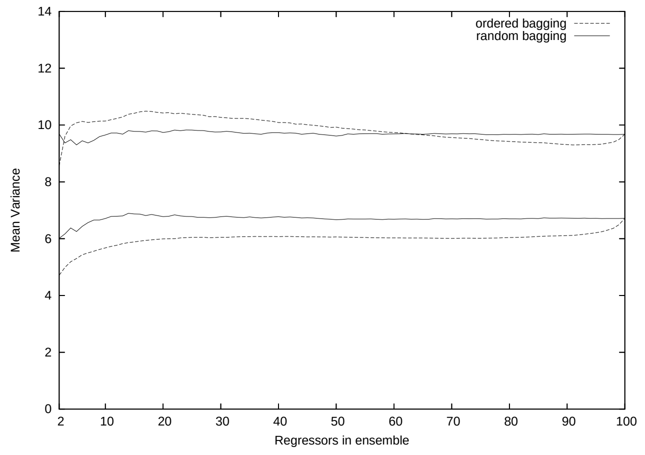

Fig. 2. Ordered bagging and randomly ordered bagging average train and test variance of the members of the ensemble, for Boston data set, as a function of the number of regressors in the ensemble.

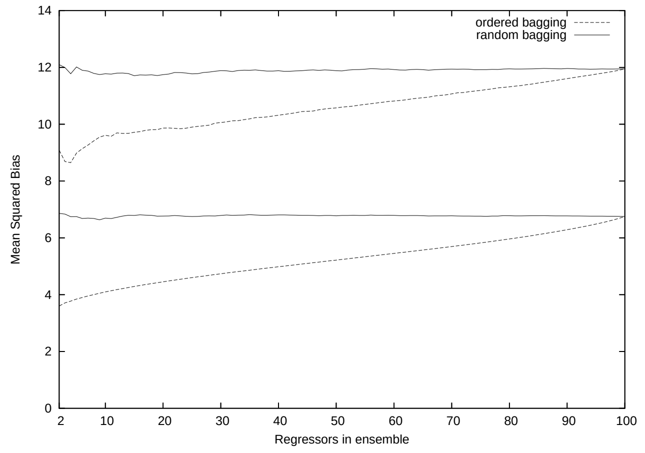

Fig. 3. Ordered bagging and randomly ordered bagging average train and test squared bias value for Boston data set, as a function of the number of regressors in the ensemble.

Results are 10 fold cross-validation estimates averaged over 10 different partitions of the data. Results are shown for both ordered and random ordered bagging. For a subensemble of size $u \geq 2$ the average values over the dataset Z of squared bias and the variance are given by the unbiased estimates

$$\widehat{Bias}^2(u;Z) = u^{-1}(u-1)^{-1}\sum_{i=1}^{u}\sum_{j=1;j\neq i}^{u} C_{ij}(Z)$$
$$\widehat{Var}(u;Z) = u^{-1}\sum_{i=1}^{u} C_{ii}(Z) - \widehat{Bias}^2(u)(Z)$$

where

$$C_{ij}(Z) = |Z|^{-1}\sum_{x\in Z}(\hat{f}_i(x) - f(x))(\hat{f}_j(x) - f(x)).$$

As anticipated, for randomly ordered bagging the values of the variance and of the squared bias remain approximately constant as the number of regressors in the ensemble is increased. The deviations from a constant behavior in the curves for the variance in small subensembles are probably due to sampling fluctuations. In agreement with the large u limit of Eq. (7) the bias term for random ordered bagging is approximately equal to the generalization error of the complete bagging ensemble. In contrast to this, the variance and the squared bias values are not constant for ordered bagging. The values of the variance and of the squared bias averaged over the training data both start low and then progressively grow until they reach the constant level of randomly ordered bagging when all the regressors are included in the ensemble. The ordering algorithm works by aggregating in the ensemble first regressors that have low variance and low bias. In ordered bagging ensembles the behavior of the curves for the average squared bias is similar when the averages are calculated over the training data or over the test set. However, the curves for the average variance are very different. When the training set is used, the ordering algorithm succeeds in finding a subensemble of regressors with a low variance. This improvement is not reflected in the variance curves estimated with the test set. We believe this is due to some overfitting to the training data.

Note that, according to (7) and (8), as the size of the ensemble grows, the most important term in the error decomposition is the value of the squared bias. Hence, it seems that the algorithm decreases the generalization error mainly by reducing the bias of the regressors included in the ensemble while keeping their variance relatively low.

V. Experimental Results

To assess the performance of pruning in ordered bagging ensembles, experiments are carried out over 14 regression problems from the UCI-Repository [15], from the Weka Data Mining Tool [16] and from other sources [17], [18], [19], [20]. These datasets have been selected in order to include a wide variety of real world and synthetic regression problems. Table I displays the number of instances, the number of attributes and the source of the different datasets. For the synthetic data sets *Friedman1*, *Friedman2* and *Friedman3*, the standard deviations for the additive noise are $\sigma = 1, \sigma = 150$ and $\sigma = 0.1$, respectively. In *Friedman1, Friedman2, Friedman3*, and *Peak* the figures reported are averages over 100 independent realizations of the synthetic regression problems. For each of these problems a training set with 200 instances, and a test set of 2000 instances were generated. For the real world problems the figures given are averages of 10-fold cross-validation error estimates over ten different partitions of the data. As base learners we use feed-forward neural networks with a single hidden layer of sigmoidal neurons, and linear units at the output layer. Networks are trained over 1000 epochs using the quasi-Newton optimization method BFGS [21]. In order to avoid overfitting, a weight decay strategy is used [22] in the training process. All computations are carried out using the R statistics software [23] with the available neural networks package [24].

TABLE I

CHARACTERISTICS OF THE DATASETS USED IN THE EXPERIMENTS.

Dataset	**Cases**	**Attr.**	**Train/Test**	**Source**
Boston	506	13	10-fold-cv	UCI-Repository
Servo	167	4	10-fold-cv	UCI-Repository
Ozone	330	8	10-fold-cv	UCI-Repository
Friedman1	2200	10	200 / 2000	See [18]
Friedman2	2200	4	200 / 2000	”
Friedman3	2200	4	200 / 2000	”
Peak	2200	20	200 / 2000	See [20]
AutoPrice	159	15	10-fold-cv	Weka
Pollution	60	15	10-fold-cv	Weka
Solder	720	5	10-fold-cv	Ref. [19]
Sensory	576	11	10-fold-cv	Weka
Bodyfat	252	14	10-fold-cv	Weka
Bolts	40	7	10-fold-cv	Weka
Loblolly	84	2	10-fold-cv	See [17]

For each of the real world datasets, we use 10-fold cross-validation to estimate the regression error. This cross-validation process is repeated 10 times for different partitions of the data. The error values reported are averages over these different partitions. The computation of each cross-validation estimate involves the following steps: (i) Generate a random partition of the original data into 10 different sets in order to carry out 10-fold cross-validation. (ii) For each of the ten divisions into train and test, generate either 100 or 200 neural networks from the training set using bootstrap sampling. For each network, different architectures (2, 3, 5, and 7 hidden units) are explored. We also consider 12 possible values, equally spaced in the interval $[0, 1]$, for the weight decay constant. All possible combinations of the number of hidden units and of the values of the weight decay constant are tried to determine which model provides the best regression fit, based on a separate 10-fold cross-validation estimate of the mean square error on the training data. Once the best parameter combination is found we build the network by training over 1000 epochs. For each of the ten ensembles the error is estimated with the corresponding unseen data. Finally, these values are averaged to compute the cross-validation error. (iii) The ensembles are then ordered according to the greedy algorithm described in Section IV, using error estimates only on the training set. (iv) The first 20% regressors are selected to construct the pruned subensemble. Then, the error of each of the pruned ensembles is estimated with the corresponding unseen data. Finally, these values are averaged to compute the cross-validation estimate.

For the synthetic datasets the individual neural networks and the ensembles are generated and ordered using the procedure described in the previous paragraph. However, instead of averages of cross-validation error estimates, the figures reported are averages 100 independent realizations of the train and test datasets.

The pruning rule that selects the first 20% regressors in the ordered pool generated by bagging selects subensembles that exhibit a consistently good generalization performance in the regression problems investigated. In fact, because the test error curves are rather flat around the minimum, any value between 15% and 25% percent of the original pool of regressors leads to similar results. Furthermore, any subensemble that includes more than 15% of the initial regressors outperforms the complete bagging ensemble in all the problems investigated.

Table II shows the results for the different datasets using ensembles of 100 neural networks. The figures in first column correspond to the average training error for the complete ensemble. The second column displays the averaged training error values for the pruned ensemble. The next two columns display the same information for the errors on the test set. Table III shows the corresponding figures using ensembles of 200 neural networks. These tables show that the error of the pruned subensembles is systematically lower than the error of the complete bagging ensemble. A paired *t-test* was carried out to determine whether these improvements in the generalization accuracy are statistically significant. Table IV shows the *p-values* given by the test for each regression problem. In the synthetic regression problems (*Peak*, *Friedman1*, *Friedman2* and *Friedman3*), for ensembles of 100 members and 200 members, these p-values show that pruned subensembles have a better prediction accuracy with a confidence level of 99.9%. A similar conclusion can be reached for the real-world problems (except for the *Solder* problem, where only a 95% confidence level is achieved). However, in this case, the statistical test may overestimate the significance of the differences, because the sets used in the cross-validation estimates are not independent [25].

VI. CONCLUSIONS

In this paper we have proved that the selection of an optimal subensemble from a bagging regression ensemble that minimizes the error over a given data set is an NP-hard optimization problem. An approximate greedy strategy that generally reaches a near-optimal solution consists in ordering the regressors in the initial bagging ensemble, and then selecting a subset for aggregation. The ordering procedure incorporates first those regressors from the initial pool of classifiers that improve the regression accuracy of

TABLE II

AVERAGE TEST AND TRAIN MEAN SQUARE ERROR FOR THE COMPLETE AND PRUNED BAGGING ENSEMBLES USING A POOL OF 100 NEURAL NETWORKS. RESULTS HAVE BEEN MULTIPLIED BY A FACTOR (*) 10^{-3}, (**) 10^{3}, (***) 10^{-6}.

	Train		**Test**	
Dataset	**Complete**	**Pruned**	**Complete**	**Pruned**
Boston	6.82±0.80	4.75±0.59	12.05±1.07	10.38±0.86
Servo	0.073±0.010	0.040±0.006	0.273±0.054	0.218±0.076
Ozone	13.79±0.73	11.56±0.68	17.73±0.46	17.26±0.45
Friedman1	1.14±0.31	0.90±0.31	2.99±0.38	2.64±0.37
Friedman2*	17.40±5.14	12.07±2.35	23.01±4.52	19.56±1.27
Friedman3**	12.52±2.63	9.96±2.01	18.27±3.08	17.55±2.47
Peak	4.07±0.18	2.71±0.13	25.79±1.06	23.55±1.01
AutoPrice***	4.29±0.19	2.02±0.12	7.07±0.31	5.77±0.55
Pollution*	2.71±0.22	1.10±0.19	3.07±0.08	2.20±0.24
Solder	4.40±0.26	4.00±0.25	7.53±0.32	7.43±0.29
Sensory**	366.2±6.5	343.3±6.2	535.9±7.0	522.9±7.8
Bodyfat	1.77±0.27	0.53±0.03	2.93±0.31	1.98±0.19
Bolts	2.45±0.41	0.69±0.15	10.95±3.07	7.90±2.91
Loblolly	0.74±0.69	0.12±0.02	1.68±0.89	0.91±0.19

TABLE III

AVERAGE TEST AND TRAIN MEAN SQUARE ERROR OF COMPLETE AND PRUNED BAGGING ENSEMBLES USING 200 NEURAL NETWORKS. RESULTS HAVE BEEN MULTIPLIED BY A FACTOR (*) 10^{-3}, (**) 10^{3}, (***) 10^{-6}.

	Train		**Test**	
Dataset	**Complete**	**Pruned**	**Complete**	**Pruned**
Boston	6.80±0.75	4.61±0.60	12.04±1.02	10.14±0.87
Servo	0.071±0.009	0.038±0.006	0.270±0.055	0.210±0.067
Ozone	17.70±0.36	17.24±0.24	13.75±0.72	11.42±0.71
Friedman1	1.14±0.31	0.88±0.29	2.99±0.38	2.61±0.39
Friedman2*	17.17±4.86	11.88±2.34	22.78±4.22	19.33±1.25
Friedman3**	12.52±2.57	9.89±1.20	18.25±3.10	17.40±2.58
Peak	4.01±0.18	2.51±0.13	25.72±1.04	23.01±1.00
AutoPrice***	4.27±0.17	1.89±0.09	7.11±0.33	5.72±0.34
Pollution*	2.72±0.22	1.05±0.19	3.05±0.10	2.08±0.22
Solder	4.39±0.27	3.96±0.26	7.51±0.31	7.41±0.27
Sensory**	365.5±6.8	339.9±6.7	535.9±6.7	520.2±7.4
Bodyfat	1.73±0.23	0.51±0.03	2.87±0.28	1.93±0.19
Bolts	2.38±0.40	0.62±0.15	10.93±3.08	7.72±2.85
Loblolly	0.66±0.54	0.11±0.02	1.51±0.65	0.88±0.17

the ensemble the most. In this manner we build a sequence of subensembles of increasing size and plot error curves that display the dependence of the regression error on the subensemble size. For standard bagging, where the order of aggregation is random, these curves show a monotonic decrease of the error as the number of regressors in the subensemble is increased. In ordered bagging the mean square error reaches a minimum value for subensembles of intermediate size. This minimum corresponds to an error that is below the error of the complete bagging ensemble. Around this minimum the error curve is fairly broad, which implies that it is easy to improve the results of bagging by early stopping in the aggregation process in ordered bagging ensembles. We take advantage of this feature in the error curves and select a pruned ensemble that includes only the first 20% elements from the ordered ensemble of regressors. The pruned ensembles consistently improve the generalization performance of the complete bagging ensemble in the regression problems investigated. The proposed algorithm can be also used to prune regression bagging ensembles composed of other types of regressors as base learners.

APPENDIX

In this appendix we prove that finding the subensemble of a regression bagging ensemble that minimizes the mean square error over a given data set is an NP-hard optimization

TABLE IV

RESULTS OF A PAIRED *t-test* FOR THE DIFFERENCES BETWEEN THE TEST ERRORS OF PRUNED AND COMPLETE ENSEMBLES.

	p-values	
Dataset	**100 members**	**200 members**
Boston	$7.35{\cdot}10^{-12}$	$1.97{\cdot}10^{-14}$
Servo	$9.6{\cdot}10^{-4}$	$4.2{\cdot}10^{-4}$
Ozone	$2.9{\cdot}10^{-4}$	$2.3{\cdot}10^{-4}$
Friedman1	$2.9{\cdot}10^{-35}$	$1.5{\cdot}10^{-37}$
Friedman2	$2.7{\cdot}10^{-12}$	$1.3{\cdot}10^{-13}$
Friedman3	$7.2{\cdot}10^{-4}$	$1.4{\cdot}10^{-4}$
Peak	$2.6{\cdot}10^{-54}$	$1.5{\cdot}10^{-71}$
AutoPrice	$1.5{\cdot}10^{-7}$	$6.0{\cdot}10^{-9}$
Pollution	$3.9{\cdot}10^{-11}$	$3.7{\cdot}10^{-13}$
Solder	$4.28{\cdot}10^{-2}$	$2.1{\cdot}10^{-2}$
Sensory	$1.4{\cdot}10^{-11}$	$4.9{\cdot}10^{-16}$
Bodyfat	$5{\cdot}10^{-8}$	$4.7{\cdot}10^{-9}$
Bolts	$5.2{\cdot}10^{-7}$	$6{\cdot}10^{-7}$
Loblolly	$6.1{\cdot}10^{-4}$	$1.2{\cdot}10^{-4}$

problem. This optimal subensemble selection problem is related to the *Subset Sum* problem, which is NP-complete [11]. The *Subset Sum* problem consists in extracting from a given set of integers $S = \{s_1, s_2, ..., s_n\}$ a subset of elements whose sum is equal to zero.

Suppose that an algorithm $\mathcal{A}$ that solves the optimal subensemble selection problem in polynomial time exists. The *Subset Sum* problem would then be solvable in polynomial time by letting each of the members of the set of integers S be the output of a regressor and assuming that zero is the target output. In this manner the set S of integer values can be seen as the output of the ensemble in a dataset with only one instance. Then, according to this expression,

$$\sum_{c \in C} c = 0 \iff \left(|C|^{-1} \sum_{c \in C} c \right)^2 = 0 \ \forall C \subset S, \, C \neq \emptyset$$

finding a subensemble with zero mean square error is equivalent to finding a subset of S whose elements add up to zero. If a subset of S with zero mean square error exists (this is the lowest possible mean square error) then algorithm $\mathcal{A}$ would be able to find it in polynomial time. This subset of S has the property that its elements add up to zero and therefore is a solution to the *Subset Sum* problem. On the other hand if a subset with mean square error greater than zero is returned, then there is no subset of S the sum of whose elements is zero.

Unless NP=P, no algorithm with the properties of $\mathcal{A}$ exists. Hence, the problem of finding the optimal subensemble of a regression bagging ensemble is at least as hard as any NP-complete problem. These types of problems are NP-hard, and unless NP=P, no polynomial time algorithm that computes a solution to any of this problems exits. Notice that this result also implies that pruning a weighted averaged regression ensemble is an NP-hard problem.

ACKNOWLEDGMENT

The authors thank Jose Miguel Hernández-Lobato for his valuable comments, discussions and critical reading of this manuscript.

REFERENCES

[1] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[2] H. Drucker, "Improving regressors using boosting techniques," in *Proc. 14th International Conference on Machine Learning*. Morgan Kaufmann, 1997, pp. 107–115.

[3] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[4] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. Hinton, "Adaptive mixtures of local experts," *Neural Computation*, vol. 3, no. 1, pp. 79–87, 1991.

[5] B. Efron and R. J. Tibshirani, *An Introduction to the Bootstrap*. Chapman & Hall/CRC, 1994.

[6] Z.-H. Zhou, J. Wu, and W. Tang, "Ensembling neural networks: Many could be better than all," *Artificial Intelligence*, vol. 137, no. 1-2, pp. 239–263, 2002.

[7] Y. Liu and X. Yao, "Ensemble learning via negative correlation," *Neural Netw.*, vol. 12, no. 10, pp. 1399–1404, 1999.

[8] G. Brown, J. L. Wyatt, and P. Tino, "Managing diversity in regression ensembles," *Journal of Machine Learning Research*, vol. 6, pp. 1621–1650, 2005.

[9] S. Hashem, *Combining Articial Neural Nets*. Springer-Verlag, 1999, ch. 5, pp. 101–125.

[10] B. Bakker and T. Heskes, "Clustering ensembles of neural network models," *Neural Networks*, vol. 16, no. 2, pp. 261–269, Mar. 2003.

[11] T. H. Cormen, C. H. Leiserson, and R. L. Rivest, *Introduction to Algorithms*. The MIT Press, 1990.

[12] D. D. Margineantu and T. G. Dietterich, "Pruning adaptive boosting," in *Proc. 14th International Conference on Machine Learning*. Morgan Kaufmann, 1997, pp. 211–218.

[13] G. Martínez-Muñoz and A. Suárez, "Aggregation ordering in bagging," in *Proc. of the IASTED International Conference on Artificial Intelligence and Applications*. Acta Press, 2004, pp. 258– 263.

[14] N. Ueda and R. Nakano, "Generalization error of ensemble estimators," in *Proceedings of International Conference on Neural Networks*, 1996, pp. 90–95.

[15] C. L. Blake and C. J. Merz, "UCI repository of machine learning databases," 1998. [Online]. Available: http://www.ics.uci.edu/∼mlearn/MLRepository.html

[16] I. H. Witten and E. Frank, *Data Mining: Practical machine learning tools and techniques*, 2nd ed. San Francisco: Morgan Kaufmann, 2005.

[17] F. H. Kung, "Fitting logistic growth curve with predetermined carrying capacity." Proceedings of the Statistical Computing Section, American Statistical Association, 1986, pp. 340–343.

[18] J. H. Friedman, "Multivariate adaptive regression splines," *The Annals of Statistics*, vol. 19, no. 1, pp. 1–67, Mar. 1991.

[19] J. M. Chambers and T. J. Hastie, *Statistical Models in S*. London: Chapman & Hall, 1992.

[20] L. Breiman, "Using adaptive bagging to debias regressions," Statistics Department, University of California, Tech. Rep., 1999.

[21] J. Nocedal and S. J. Wright, *Numerical Optimization*. Springer, 1999.

[22] A. Krogh and J. A. Hertz, "A simple weight decay can improve generalization," in *Advances in Neural Information Processing Systems*, J. E. Moody, S. J. Hanson, and R. P. Lippmann, Eds., vol. 4. Morgan Kaufmann Publishers, Inc., 1992, pp. 950–957.

[23] R Development Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2005, ISBN 3-900051-07-0.

[24] W. N. Venables and B. D. Ripley, *Modern Applied Statistics with S*, 4th ed. New York: Springer, 2002, iSBN 0-387-95457-0.

[25] Y. Bengio and Y. Grandvalet, "No unbiased estimator of the variance of k-fold cross-validation." *Journal of Machine Learning Research*, vol. 5, pp. 1089–1105, 2004.