Neurocomputing 74.12-13 (2011): 2250 – 2264

DOI: http://dx.doi.org/10.1016/j.neucom.2011.03.001

Empirical Analysis and Evaluation of Approximate Techniques for Pruning Regression Bagging Ensembles

Daniel Hernández-Lobato[*,a], Gonzalo Martínez-Muñoz[b], Alberto Suárez[b]

[a] *Machine Learning Group, ICTEAM Institute, Université catholique de Louvain, Place Sainte Barbe 2, B-1348 Louvain-la-Neuve, Belgium.*
[b] *Computer Science Department, Escuela Politécnica Superior, Universidad Autónoma de Madrid, C/ Francisco Tomás y Valiente, 11, Madrid 28049 Spain.*

Abstract

Identifying the optimal subset of regressors in a regression bagging ensemble is a difficult task that has exponential cost in the size of the ensemble. In this article we analyze two approximate techniques especially devised to address this problem. The first strategy constructs a relaxed version of the problem that can be solved using Semidefinite Programming. The second one is based on modifying the order of aggregation of the regressors. Ordered Aggregation is a simple forward selection algorithm that incorporates at each step the regressor that reduces the training error of the current subensemble the most. Both techniques can be used to identify subensembles that are close to the optimal ones, which can be obtained by exhaustive search at a larger computational cost. Experiments in a wide variety of synthetic and real-world regression problems show that pruned ensembles composed of only 20% of the initial regressors often have better generalization performance than the original bagging ensembles. These improvements are due to a reduction in the bias and the covariance components of the generalization error. Subensembles obtained using either SDP or Ordered Aggregation generally outperform subensembles obtained by other ensemble pruning methods and ensembles generated by the Adaboost.R2 algorithm, negative correlation learning or regularized linear stacked generalization. Ordered Aggregation has a slightly

[*]Corresponding author. Tel: +32-10-47-2445; fax: +32-10-45-0345.
Email addresses: daniel.hernandez-lobato@uclouvain.es (Daniel Hernández-Lobato), gonzalo.martinez@uam.es (Gonzalo Martínez-Muñoz), alberto.suarez@uam.es (Alberto Suárez)

better overall performance than SDP in the problems investigated. However, the difference is not statistically significant. Ordered Aggregation has the further advantage that it produces a nested sequence of near-optimal subensembles of increasing size with no additional computational cost.

Key words: Regression, Ensemble Learning, Bagging, Boosting, Semidefinite Programming, Ensemble Pruning

1. Introduction

Ensembles have the potential to improve the performance of an individual predictor by combining the outputs of a collection of complementary predictors. A widely used algorithm to build ensembles for classification and regression is bagging [6]. In bagging, different models are generated by applying the same learning algorithm to independent bootstrap samples of the original training data. If the learning algorithm used is unstable (that is, if small changes in the training set lead to different models), a collection of diverse regressors can be induced from the different bootstrap samples. If the errors of these elements are uncorrelated, improved predictions can be obtained by averaging the outputs of the regressors in the ensemble. This mechanism generally reduces the variance component of the generalization error [7, 8, 49]. Bagging is a very robust learning algorithm, which can perform well even when data are noisy [17, 41]. Furthermore, including more elements in a bagging ensemble does not generally lead to overfitting [11]. Typically, the prediction error decreases monotonically with the ensemble size, and asymptotically approaches a constant level, which is considered the best result that bagging can achieve.

Another popular ensemble algorithm is boosting. Boosting was originally developed for classification problems [19, 46]. In boosting, a sequence of models is induced from a given dataset using a fixed learning algorithm and different weights for the different training instances. The first model in the sequence is learned using equal weights. The $(n+1)$th model in the ensemble is constructed by modifying the weights of the instances used for induction: Larger weights are assigned to training examples that are incorrectly labeled by the nth model in the sequence. By contrast, the weights of correctly labeled instances are lowered. The final prediction of the boosting ensemble is a weighted combination of the predictions of the individual models. The weights in this combination are determined by the error on the training

data: The lower the training error of the model, the higher its weight in the ensemble prediction and vice versa. Boosting has been shown to perform well in several benchmark classification problems [3, 9, 41]. However, it is very sensitive to noise in the class labels and its effectiveness is severely reduced in noisy domains [17, 41]. Boosting has also been adapted to solve regression problems [1, 18, 19, 21]. In this work we use the Adaboost.R2 algorithm, one of the extensions of *Adaboost* for regression proposed in [18]. This variant of boosting has been selected because of its good performance in several regression problems [18]. Furthermore, Adaboost.R2 is used as a benchmark for comparison in previous work on regression ensembles [12, 56, 1].

An important shortcoming of ensemble methods is that, in many problems of practical interest, many predictors are needed to achieve good generalization performance. Large ensembles require more storage space and take longer to make predictions. A possible way to alleviate this problem is to replace the original ensemble by a representative subset of predictors. This approach has been shown to be effective in classification problems: Pruned subensembles can in fact outperform the original ensembles from which they are extracted [2, 13, 30, 33, 34, 35, 55, 31]. Nevertheless, the task of selecting from a pool of predictors a subset whose performance is optimal is a difficult problem. On the one hand, it is computationally expensive: The search is conducted in the space of $2^M - 1$ non-empty subensembles that can be extracted from an ensemble of size M. On the other hand, even if the search were feasible, the selection is made in terms of an objective function estimated on the training data. Since we are interested in out-of-sample performance, finding the optimum in the training set does not guarantee optimal generalization properties.

Previous studies have shown that the generalization performance of regression bagging ensembles can be improved by selecting a subset of complementary regressors whose prediction errors are uncorrelated [39, 56, 26]. These techniques can be used in principle to prune other types of ensembles in which the final prediction is obtained by averaging over the individual predictors that make up the ensemble [44, 54, 37, 32]. The main goals of this research are to design extensions to regression problems of pruning techniques that have been previously introduced in the context of classification problems, to analyze their properties and to evaluate their performance. The first technique investigated is an adaptation of the pruning method based on Semi-definite Programming (SDP) introduced in [55]. An alternative approach to ensemble pruning consists in modifying the order of aggregation in

the ensemble [31]. Starting with an empty subensemble, Ordered Aggregation follows a forward selection strategy and incorporates the regressor that reduces the training error of the current subensemble the most, until a stopping criterion is met. In the current work we show that the subensembles identified by the ordering procedure are very similar both in performance (quantified in terms of either the training error or the test error) and in composition to optimal subensembles of the same size, identified by exhaustive search. The optimality of the subensembles is determined in terms of the training set alone, which are the only data available for induction.

As noted earlier, minimizing a measure of performance on the training set does not necessarily lead to improvements in generalization capacity. To evaluate the generalization performance of the pruned subensembles and to compare it with other ensemble generation and ensemble-pruning methods, we carry out extensive experiments in 24 synthetic and real-world datasets. The results of these experiments show that SDP-pruning and Ordered Aggregation are effective methods for identifying subensembles with good generalization performance not only in classification problems, but also in regression tasks. A detailed analysis of the components of the generalization error of the subensembles confirms that the improvements in prediction accuracy arise because both SDP-pruning and Ordered Aggregation select from the original ensembles subsets of regressors that simultaneously have a low bias (i.e. their individual errors tend to be low) and small or negative correlations (i.e. their predictions are complementary).

In the problems investigated, and for a large range of pruning rates, pruned subensembles obtained with these two methods outperform single neural networks, complete ensembles built either with bagging or Adaboost.R2 and other pruning techniques based on genetic algorithms [56]. The differences observed are statistically significant. The performance of the subensembles identified via SDP-pruning or by Ordered Aggregation is comparable with the subensembles identified by other pruning methods described in the literature [39] and with the ensembles generated by negative correlation learning [29] or regularized linear stacked generalization [43]. The overall performance of Ordered Aggregation is slightly better than SDP. However, the difference is too small to be statistically significant. The simple forward selection used in Ordered Aggregation has the advantage of producing a nested sequence of near-optimal subensembles of increasing size. By contrast, SDP requires to solve a different semi-definite programming problem for each ensemble size, which entails a correlative increase in computational costs.

The article is organized as follows: Section 2 introduces the problem of selecting an optimal subensemble from a pool of regressors generated with bagging. Since this problem is NP-hard, approximate methods have to be used in practice. Near-optimal solutions can be found by SDP-pruning or by Ordered Aggregation. An empirical analysis of these techniques is presented in Section 3. Specifically, Section 3.1 shows that the subensembles identified using either SDP-pruning or Ordered Aggregation are very close to the optimal subensembles obtained by exhaustive search. These near-optimal subensembles actually outperform the original complete ensembles from which they are extracted. The bias-variance-covariance analysis presented in Section 3.2 elucidates the origin of the improvements obtained. The results of the extensive assessment of the performance of the pruning methods analyzed are presented in Section 3.3. Finally, Section 4 summarizes the results and conclusions of this work.

2. Selection of Optimal Subensembles

Consider a regression problem. The goal is to learn a function that predicts the dependent variable $y \in \mathbb{R}$ in terms of the attributes $\mathbf{x} \in \mathcal{X}$ using a set of training data $\mathcal{D} = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\}$, whose instances are independently drawn from a probability distribution $\mathcal{P}(\mathbf{x}, y)$. Bagging works by combining the prediction of a collection of regressors. Each of these regressors is constructed by applying a fixed learning algorithm on a different bootstrap sample from the original training data $\mathcal{D}$. Let $\hat{f}_i(\mathbf{x})$ be the prediction given by the ith regressor built with $\mathcal{D}_i$, the ith bootstrap sample from the training data. The prediction of the ensemble is the average of the individual predictions of the M regressors in the ensemble

$$M^{-1} \sum_{i=1}^{M} \hat{f}_i(\mathbf{x}), \ i = 1, 2, \ldots, M. \tag{1}$$

The generalization error of the ensemble is

$$\mathcal{L} = \int \left(M^{-1} \sum_{i=1}^{M} \hat{f}_i(\mathbf{x}) - y \right)^2 \mathcal{P}(\mathbf{x}, y) d\mathbf{x} dy, \tag{2}$$

where $y = f(\mathbf{x})$, f is the target function to approximate, and $\mathcal{P}(\mathbf{x}, y)$ is the probability distribution of the data. After some algebra (2) can be expressed

as

$$\mathcal{L} = M^{-2} \sum_{i=1}^{M} \sum_{j=1}^{M} \mathcal{C}_{ij} \tag{3}$$

where

$$\mathcal{C}_{ij} = \int \left(\hat{f}_i(\mathbf{x}) - y \right) \left(\hat{f}_j(\mathbf{x}) - y \right) \mathcal{P}(\mathbf{x}, y) d\mathbf{x} dy \,. \tag{4}$$

The value $\mathcal{C}_{ii}$ is the average squared error of the ith ensemble member. The off-diagonal terms $\{\mathcal{C}_{ij},\ i \neq j\}$ correspond to correlations between the predictions of the ith and jth ensemble members [56, 39].

Consider a bagging ensemble of size M. Our goal is to select the subensemble of u regressors $\{s_1, s_2, \ldots, s_u\}$ that minimizes the error

$$\mathcal{L}^{(u)} = u^{-2} \sum_{i=1}^{u} \sum_{j=1}^{u} \mathcal{C}_{s_i s_j}, \tag{5}$$

where $\{s_1, s_2, \ldots, s_u\}$ is a subset of the indices that label the ensemble members $\{1, 2, \ldots, M\}$. Since the actual generalization error cannot be computed, the selection of the optimal subensemble is made on the basis of the training error. The expression for the training error is identical to (5), except that the average over $\mathcal{P}(\mathbf{x}, y)$ in the calculation of $\mathcal{C}_{ij}$ is replaced by a sample average over the training data

$$\mathcal{C}_{ij}^{(tr)} = \frac{1}{N} \sum_{n=1}^{N} \left(\hat{f}_i(\mathbf{x}_n) - y_n \right) \left(\hat{f}_j(\mathbf{x}_n) - y_n \right) . \tag{6}$$

Hence, the information needed for the optimization problem is contained in the matrix $\boldsymbol{\mathcal{C}}^{(tr)}$, which is estimated on the training set. This estimate is expected to be close to the true $\boldsymbol{\mathcal{C}}$ matrix, which is calculated as an average over the actual distribution of the data, so that minimizing the training error leads to the minimization of the generalization error. This is not necessarily the case in actual regression problems: minimizing the training error sometimes leads to overfitting, and, consequently, to the selection of subensembles whose generalization performance is suboptimal. In the experiments performed, overfitting becomes manifest in the fact that, typically, the subensembles that minimize the error on the training data tend to be smaller than the subensembles that are optimal when the error is estimated on an independent test set.

In any case, even if it were possible to accurately predict the generalization error from the training data only, finding the optimal subensemble is a computationally expensive problem that involves comparing the performance of all the possible $2^M - 1$ non-empty subensembles that can be extracted from the original ensemble. In fact, the problem of selecting the optimal subensemble from a given ensemble is NP-hard (see Appendix A). This implies that finding the exact solution is infeasible for large ensembles. In this work we analyze two techniques that can identify near-optimal subensembles at a reduced computational cost[1]. The first method solves a relaxed version of the problem using semidefinite programming (SDP). The second one is based on modifying the order of aggregation of the regressors in the ensemble, which in standard bagging is random.

2.1. Ensemble pruning via semidefinite programming

The method based on Semidefinite programming (SDP) introduced in [55] by Zhang *et al.* to prune classification ensembles can be extended to ensembles of regressors. In classification tasks the subensemble selection problem is formulated in terms of an $M \times M$ matrix $\mathbf{G}$, whose diagonal terms G_{ii} measure the individual training errors of the ensemble members and whose off-diagonal terms G_{ij}, $i \neq j$ measure the number of common training errors between classifiers i and j. The goal is to find the sub-matrix of $\mathbf{G}$, of dimensions $u \times u$, that corresponds to a subensemble of size u that minimizes the sum of the elements in $\mathbf{G}$. This process need not provide the subensemble of size u with the minimum training error. However, it guarantees that the subensemble elements have small training error rates and that they tend to make incorrect predictions for different training instances.

The problem described in the previous paragraph is a standard 0-1 optimization problem that is also NP-hard. Zhang *et al.* reformulate this problem so that it has the same optimal solutions as an instance of the max-cut problem of size u (MC-u). This problem consists in partitioning the vertices of an edge-weighted graph into two sets, one of which has as size u, so that the total weight of edges crossing the partition is maximized. Good approximate solutions to the max-cut problem can be obtained using an algorithm based on SDP [23, 24]. Therefore, good approximations to the problem of selecting

[1]Source code available for both pruning techniques at `http://www.eps.uam.es/~gonzalo/publications/`

an optimal subensemble of size u can be found by solving the corresponding instance of the MC-u problem.

The ensemble pruning problem of Zhang *et al.* is formulated in [55] as

$$\begin{array}{ll} \min\limits_{\mathbf{u}} & \mathbf{u}^T\mathbf{G}\mathbf{u} \\ \text{s.t.} & \sum_i u_i = u, \\ & u_i \in \{0,1\}, \end{array} \tag{7}$$

The binary variable u_i takes the value 1 if ith predictor is selected and 0 if it is not selected. Without the cardinality constraint, $\sum_i u_i = u$, there is a trivial solution to the problem, in which none of the regressors is selected. They then propose the transformation $u_i = (v_i+1)/2$ with $v_i \in \{-1,1\}$. With this change of variables, the objective function is proportional to $(\mathbf{v}+\mathbf{e})^T\mathbf{G}(\mathbf{v}+\mathbf{e})$, where $\mathbf{e}$ is a column vector composed of M ones. The constraint $\sum_i u_i = u$ becomes $(\mathbf{v}+\mathbf{e})^T\mathbf{I}(\mathbf{v}+\mathbf{e}) = 4u$, where $\mathbf{I}$ is the identity matrix. Defining the matrices

$$\begin{array}{rcl} \mathbf{H} & = & \begin{pmatrix} \mathbf{e}^T\mathbf{G}\mathbf{e} & \mathbf{e}^T\mathbf{G} \\ \mathbf{G}\mathbf{e} & \mathbf{G} \end{pmatrix}, \\ \mathbf{D} & = & \begin{pmatrix} M & \mathbf{e}^T \\ \mathbf{e} & \mathbf{I} \end{pmatrix}, \end{array} \tag{8}$$

and extending the vector $\mathbf{v}$ with one additional component v_0 that takes value one, (7) becomes

$$\begin{array}{ll} \min\limits_{\mathbf{v}} & \mathbf{v}^T\mathbf{H}\mathbf{v} \\ \text{s.t.} & \mathbf{v}^T\mathbf{D}\mathbf{v} = 4u, \\ & v_0 = 1, \\ & v_i \in \{-1,1\}\,\forall i \neq 0. \end{array} \tag{9}$$

This problem is equivalent to

$$\begin{array}{ll} \min\limits_{\mathbf{v}} & \mathbf{H} \bullet \mathbf{v}\mathbf{v}^T \\ \text{s.t.} & \mathbf{D} \bullet \mathbf{v}\mathbf{v}^T = 4u, \\ & v_0 = 1, \\ & v_i \in \{-1,1\}\,\forall i \neq 0. \end{array} \tag{10}$$

where $\mathbf{A} \bullet \mathbf{B} = \sum_{ij} A_{ij}B_{ij}$. This latter formulation of the problem is equivalent to the MC-u problem [55]. The goal is now to construct the relaxed version of the problem that can be efficiently solved using SDP. The constraint $v_0 = 1$ in (10) is relaxed to $v_0 \in \{-1,1\}$ without modifying the

problem, because $-\mathbf{v}$ is feasible whenever $\mathbf{v}$ is feasible. Next, the constraints $v_i \in \{-1, 1\}$ are rewritten in the form $\text{diag}(\mathbf{v}\mathbf{v}^T) = \mathbf{e}$

$$\begin{aligned} \min_{\mathbf{v}} \quad & \mathbf{H} \bullet \mathbf{v}\mathbf{v}^T \\ \text{s.t.} \quad & \mathbf{D} \bullet \mathbf{v}\mathbf{v}^T = 4u, \\ & \text{diag}(\mathbf{v}\mathbf{v}^T) = \mathbf{e}. \end{aligned} \tag{11}$$

The problem can be expressed in terms of a positive semidefinite matrix $\mathbf{V}$ of rank one

$$\begin{aligned} \min_{\mathbf{V}} \quad & \mathbf{H} \bullet \mathbf{V} \\ \text{s.t.} \quad & \mathbf{D} \bullet \mathbf{V} = 4u, \\ & \text{diag}(\mathbf{V}) = \mathbf{e}, \\ & \mathbf{V} \succeq 0. \\ & \text{rank}(\mathbf{V}) = 1. \end{aligned} \tag{12}$$

This reformulation is possible because $\mathbf{v}\mathbf{v}^T = \mathbf{V}$ if and only if $\mathbf{V} \succeq 0$ and $\text{rank}(\mathbf{V}) = 1$.

From this reformulation it is possible to obtain a convex SDP optimization problem from (12) by dropping the rank constraint

$$\begin{aligned} \min_{\mathbf{V}} \quad & \mathbf{H} \bullet \mathbf{V} \\ \text{s.t.} \quad & \mathbf{D} \bullet \mathbf{V} = 4u, \\ & \text{diag}(\mathbf{V}) = \mathbf{e}, \\ & \mathbf{V} \succeq 0. \end{aligned} \tag{13}$$

This SDP problem can be efficiently solved in polynomial time with a suitable optimizer, such as the one designed in [5]. As described in [55], from a solution matrix $\mathbf{V}$ of (13), a solution vector $\mathbf{u}$ is obtained by a randomized approximate algorithm [23, 24]. Specifically, the components of $\mathbf{u}$ are determined by sampling $\mathbf{v}$ from a Gaussian distribution $\mathcal{N}(0, \mathbf{V})$ and then applying the rule

$$u_i = \begin{cases} 1 & \text{if } \text{sign}(v_i) = \text{sign}(v_0) \\ 0 & \text{if } \text{sign}(v_i) \neq \text{sign}(v_0) \end{cases}. \tag{14}$$

If the targeted number of regressors in the subensemble is not correctly determined by the application of this rule, Zhang *et al.* use a greedy algorithm that incorporates or removes elements in $\mathbf{u}$, as needed, causing the least deterioration in (7). This procedure is repeated ten times and the best solution is retained.

Even though (13) can be cast in a form that is equivalent to the MC-u problem, the approximation bounds that hold for the relaxed version of this problem [23, 24] are not applicable in the relaxed version of subset selection. Subset selection and the MC-u problem share optimal solutions but not optimal values. The reason is that the objective function in (13) does not exactly match the objective in the MC-u problem [55]. Despite this lack of guarantees for the quality of the approximation, the procedure described is very effective in selecting near-optimal ensembles in classification tasks [55, 31].

The approach described above for pruning classification ensembles can be extended to prune regression ensembles. For this, we observe that the generalization error of the initial bagging ensemble can be expressed in terms of the matrix $\boldsymbol{\mathcal{C}}$ in (3). Thus, the subensemble selection problem consists in finding a subensemble of size u for which the sum of the elements in the corresponding sub-matrix of $\boldsymbol{\mathcal{C}}$ is as low as possible. An approximate solution to this problem can be obtained using the method of Zhang *et al.* to prune classification ensembles. The difference lies in the matrix that is used to guide the optimization process. In [55] it is the matrix $\mathbf{G}$. In the regression case, this matrix is replaced by an appropriate estimate of the matrix $\boldsymbol{\mathcal{C}}$, the matrix $\boldsymbol{\mathcal{C}}^{(tr)}$, defined in (6). The diagonal elements of this matrix measure the individual squared errors on the training set, and the off-diagonal elements correspond to correlation-like values.

The computational cost of selecting a subensemble of size u by solving problem (13) is $\mathcal{O}(M^3)$, where M is the size of the initial ensemble. The computation of $\boldsymbol{\mathcal{C}}^{(tr)}$ is $\mathcal{O}(M^2 \cdot N)$. Therefore, the total cost of the SDP method for pruning regression ensembles is $\mathcal{O}(M^3 + M^2 \cdot N)$, where N is the size of the training set $\mathcal{D}$. Extracting all near-optimal subensembles of sizes $u = 1, 2, \ldots, M$, implies solving (13) M times, with an overall cost $\mathcal{O}(M^4 + M^3 \cdot N)$.

2.2. Ordered Aggregation

Another strategy that can be used to find an approximate solution to the subensemble selection problem is based on modifying the order in which the predictors are aggregated into the ensemble. This strategy has been successfully used for pruning classification ensembles [13, 30, 33, 34, 35, 45, 31]. Specifically, one of the goals of [31] was to determine the relevance of the choice of ordering heuristic to the performance of the resulting pruned ensembles. The conclusion of that study is that the effectiveness of different heuristics that take into account both the individual prediction accuracy and

the complementarity of the classifiers is very similar. The goal of the present work is to extend this analysis to regression problems, where the optimization of the training error provides a natural criterion for the selection of subensembles. The effectiveness of Ordered Aggregation in regression bagging ensembles was assessed in [26]. In the present investigation we provide a more detailed analysis of Ordered Aggregation and a more extensive evaluation, including comparisons with related ensemble creation and ensemble pruning methods, such as negative correlation learning, SDP-pruning, the pruning method proposed by Perrone and Cooper [39], the genetic algorithm described in [56] and regularized linear stacked generalization ensembles obtained using the lasso [43].

From the initial pool of M regressors generated by bagging, ordered aggregation builds a sequence of nested ensembles, in which the ensemble of size u contains the ensemble of size $u-1$. The algorithm starts with an empty ensemble that grows by incorporating at each iteration the regressor that reduces the training error of the enlarged subensemble the most. In particular, the regressor selected in the uth iteration is the one that minimizes the expression

$$s_u = \arg\min_k \; u^{-2}\left(\sum_{i=1}^{u-1}\sum_{j=1}^{u-1} \mathcal{C}^{(tr)}_{s_i s_j} + 2\sum_{i=1}^{u-1} \mathcal{C}^{(tr)}_{s_i k} + \mathcal{C}^{(tr)}_{kk}\right) \tag{15}$$

where $k \in \{1,..,N\}\backslash\{s_1, s_2..., s_{u-1}\}$ and $\{s_1, s_2..., s_{u-1}\}$ label regressors that have been incorporated in the pruned ensemble at iteration $u-1$. Fig. 1 displays the pseudo-code for Ordered Aggregation.

This process can be seen as an ordering of the regressors of the complete ensemble because the subensemble generated at iteration u includes all the regressors of the subensemble generated at iteration $u-1$. The subensemble of size u with $1 \leq u \leq M$ is obtained by taking the first u regressors from the ordered sequence. Subensembles selected by Ordered Aggregation need not be optimal. In particular, the optimal subensemble of size u (the one with the lowest mean squared error on the training data) need not include all the regressors of the optimal ensemble of size $u-1$. Nonetheless, Ordered Aggregation is expected to identify near-optimal solutions of the subensemble selection problem.

The time-complexity of this algorithm, as a function of the number of regressors in the bagging ensemble, can be readily estimated. Each of the M iterations requires the extraction of the regressor that minimizes (15) from

Input: Training data $\mathcal{D} = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\}$ with $\mathbf{x}_i \in \mathcal{X}$, $y_i \in \mathcal{Y}$ and $i = 1, ..., N$. Vector of regressors $\mathcal{R} = \{\hat{f}_1(\cdot), \ldots, \hat{f}_M(\cdot)\}$

1. **For** $i = 1, \ldots, M$:
   (a) **For** $j = 1, \ldots, M$:
      i. $$\mathcal{C}_{ij} \leftarrow N^{-1} \sum_{n=1}^{N} \left[\left(\hat{f}_i(\mathbf{x}_n) - y_n \right) \left(\hat{f}_j(\mathbf{x}_n) - y_n \right) \right]$$
2. $\mathbf{s} \leftarrow$ empty vector
3. **For** $u = 1, \ldots, M$:
   (a) min $\leftarrow +\infty$
   (b) **For** k in $\{1, \ldots, M\} \backslash \{s_1, \ldots, s_u\}$:
      i. value $\leftarrow u^{-2}(\sum_{i=1}^{u-1} \sum_{j=1}^{u-1} \mathcal{C}_{s_i s_j} + 2 \sum_{i=1}^{u-1} \mathcal{C}_{s_i k} + \mathcal{C}_{kk})))$
      ii. **if** value $<$ min
         A. $s_u \leftarrow k$
         B. min $\leftarrow$ value
4. return $\mathbf{s}$

Output: Ordered vector of regressors.

$$\mathcal{R} = \{\hat{f}_{s_1}(\cdot), \ldots, \hat{f}_{s_M}(\cdot)\}$$

Fig. 1: Pseudo-code that implements Ordered Aggregation.

the remaining pool of regressors. This task has a cost $\mathcal{O}(((M+1)-u)\cdot u)$, where $1 \leq u \leq M$ is the current iteration. Therefore, the total complexity of the ordering is $\mathcal{O}(M^3)$. Finally, because computing $\boldsymbol{\mathcal{C}}^{(tr)}$ takes $\mathcal{O}(M^2 \cdot N)$ steps the final cost is $\mathcal{O}(M^3 + M^2 \cdot N)$. In contrast to SDP-pruning, where selecting subensembles of different sizes requires separate executions of the algorithm, Ordered Aggregation generates a sequence of near-optimal subensembles of increasing size at no additional cost.

2.3. Related work

The key to the improvements in the performance of the ensemble is the complementarity of the predictions given by the individual ensemble members [12, 17, 26, 49]. Individual measures of accuracy or measures of diversity cannot be used in isolation to improve the performance of the ensemble. Successful ensemble pruning techniques need to encourage complementarity among the selected predictors.

The idea of modifying the order in which the classifiers are aggregated in the ensemble was introduced as a possible improvement of ensemble learning in [39] by Perrone and Cooper. However, the order of aggregation proposed in that work is based on the individual properties of the ensemble members (e.g. the mean squared error of each neural network). The joint performance of the predictors is considered only at a later stage, to decide whether a candidate neural network does not lead to a reduction of the subensemble error and should therefore be excluded from the current subensemble. Specifically, [39] suggests to swap such a network with the closest *un-tested* network in the ordered sequence. The pruning method stops when none of the un-tested networks in the ordered sequence provide a reduction of the subensemble error.

Building an ensemble of regressors whose predictions have small or negative correlations is one of the design goals of negative correlation learning [29]. In this learning algorithm, diversity is encouraged by simultaneously training a collection of networks using a cost function that includes a correlation penalty term in addition to the prediction error. The correlation penalty term encourages the specialization of the individual networks and cooperation among them. However, the strength of the penalty needs to be to be carefully tuned for each problem. If it is too small, the cooperation among the ensemble members will not be sufficient to produce significant improvements in performance. If the penalty is too large, the learning process is ineffective: the cost function is dominated by the penalty term and becomes

insensitive to prediction errors. A further difficulty of this method is that the correlation penalty introduces a coupling among the parameters of the different ensemble members. This coupling increases the dimensionality of the parameter space in which the search is conducted and makes the learning process more difficult.

Genetic algorithms (GAs) have also been proposed for the selection of a near-optimal subensemble from a complete bagging ensemble. In [56] the output of the ensemble is a weighted average of the outputs of each ensemble member. The optimal set of weights of the ensemble members is found by minimizing a function that estimates the generalization error of the ensemble. The minimization problem is solved by GASEN (Genetic Algorithm based Selective ENsemble), a standard GA with a floating-point encoding scheme for the real-valued weights. Once evolution has finished, the neural networks whose optimized weights are below a specified threshold are removed from the ensemble. The final ensemble output is the average over the predictions of the networks retained in the ensemble. The experiments carried out in [56] establish GA as a viable strategy to prune bagging ensembles. However, the ensembles considered in that work are rather small. In fact, for the ensemble sizes considered (20 networks) exhaustive search is still feasible. In this work we show that this GA can also be used to extract subensembles with good prediction properties from larger ensembles, such as the ones typically generated in bagging (≈ 100 learners).

Subset selection techniques [25, 36] can also be used to identify subsets of predictors that outperform the original complete ensemble. The stacked generalization method described in [43] is illustrative of this approach. In stacked generalization, the vector of predictions of the different ensemble members for each training sample is considered as a data instance in a new feature space [53]. These instances are then used to train a meta-learner that produces the final ensemble prediction. In practice, the predictor used for output combination can be built using any learning algorithm. There is extensive empirical evidence that stacking predictors tends to overfit unless the combiner function is sufficiently smooth [43]. Specifically, linear combinations generally perform better than non-linear models [48]. Nevertheless, when the number of the predictors in the ensemble is large, even linear models tend to overfit [13]. This problem can be alleviated by the application of regularization techniques [43]. If regularization is performed via the lasso [47] or the elastic net regression [57] some of the coefficients of the linear model will take value zero. Therefore, the corresponding predictors can be removed

from the ensemble, because they are not involved in the combined prediction. In contrast to the ensemble pruning methods considered in this work, which assume equal weights, the remaining predictors can have different weights in the final prediction. In this work, we analyze the performance of ensembles of this type generated via stacking combined with lasso regularization. Instead of the lasso other subset selection techniques can be used [36]. Experiments on a variety of regression problems show that variants of these subset selection techniques in which uniform weights are used (instead of the actual non-zero weights calculated) have poorer generalization performance. In consequence, these variants are not considered for further evaluation.

3. Empirical Analysis and Evaluation

In this section we carry out an empirical analysis of SDP-pruning and Ordered Aggregation and evaluate their performance. For this purpose, we report the results of experiments on a wide range of problems, which include synthetic and real-world data from different domains of application. The list of problems is displayed in Table 1. In Section 3.1, the subensembles identified by these two approximate strategies are compared with the optimal subensembles of the same size obtained by exhaustive search. From the results of these experiments one concludes that the subensembles obtained by SDP-pruning and Ordered Aggregation are near-optimal and that they can improve the prediction accuracy of the original bagging ensembles from which they are extracted. The bias-variance-covariance decomposition presented in Section 3.2 provides some insight into the mechanism by which these pruning methods improve the generalization performance. Finally, in Section 3.3 the prediction accuracy of the subensembles identified by SDP-pruning and Ordered Aggregation is quantified and compared to related methods, which have been described in Section 2.3.

3.1. Optimality of the selected subensembles

SDP-pruning and Ordered Aggregation are approximate optimization methods. Therefore, they can select subensembles that are suboptimal. Globally optimal subensembles of a given size, u, can actually be identified using exhaustive search. Therefore, we have performed a series of experiments to determine how similar are the subensembles identified by these methods and the optimal ones. Optimality is defined in terms of the training data, because those are the only instances available for induction. The

question is whether the minimization of the training data leads to improvements of the generalization performance. To elucidate this point, further comparisons are made in terms of the performance on the training data and on an independent test set. The experiments are carried out in the regression problems *Servo*, *Pollution*, *Boston* and *Solder*. The results obtained in these regression tasks are representative of the problems considered in our research.

For each dataset, 10-fold cross-validation is used to estimate the squared prediction error. This cross-validation process is repeated 10 times for different random partitions of the data. The values reported are averages of the cross-validation estimates over these different partitions. The experiments involve building 100 different bagging ensembles of $M = 32$ predictors. As base learners we use feed-forward neural networks with a single hidden layer of sigmoidal neurons and a linear unit in the output layer. The networks are trained during 1000 epochs using the quasi-Newton optimization method BFGS [38]. A weight decay strategy is used in the training process to limit the amount of overfitting [27]. Previous to the generation of the bagging ensemble, the optimal architecture of the networks and the optimal value for the weight decay constant are estimated based on a separate 10-fold cross-validation estimate on the training data. Different architectures (3, 5, and 7 hidden units) are explored. We also consider ten evenly-spaced values for the logarithm of the weight decay constant in the interval $[-6, 2]$. All possible combinations of the number of hidden units and of the values of the weight decay constant are tried to determine which set of parameters provides the best regression fit. Once the best architecture and decay constant are found, a bagging ensemble is generated using neural networks with these hyper-parameters. The computations are performed using the neural networks package [50] of the R statistics software [42].

For each ensemble, optimal subensembles of sizes $u = 1, \ldots, M$ are identified by exhaustive search. SDP-pruning and Ordered Aggregation are used to generate near-optimal subensembles of different sizes. Fig. 2 displays for each problem the percentage of regressors that appear both in the optimal subensembles and in the approximate ones (either the one identified by Ordered Aggregation or by SDP-pruning) as a function of the subensemble size $1 \leq u \leq 32$. The curves show that the subensembles identified by SDP-pruning are very similar to those obtained by exhaustive search. Ordered aggregation identifies subensembles that share on average at least 70 percent of their elements with the optimal ones.

Table 1: Characteristics of the datasets used in the experiments.

Dataset	**Cases**	**Attr.**	**Source**
Attitude	30	6	R
AutoPrice	159	15	Weka
Bodyfat	252	14	Weka
Bolts	40	7	Weka
Boston	506	13	UCI-Repository
Chick	578	3	R
Concrete Slump	103	7	UCI-Repository
Fires	517	13	UCI-Repository
Friedman1	-	10	[20]
Friedman2	-	4	[20]
Friedman3	-	4	[20]
Loblolly	84	2	[28]
Longley	16	6	R
Orange	35	2	R
Ozone	330	8	UCI-Repository
Peak	-	20	[10]
Pollution	60	15	Weka
Rock	48	3	R
Sensory	576	11	Weka
Servo	167	4	UCI-Repository
Solder	720	5	R
Theoph	132	4	R
Tooth	60	2	R
Wisconsin	198	35	UCI-Repository

Next, the prediction error of these subensembles is estimated in the training set and in an independent test set. Optimality is defined in terms of the prediction accuracy in the training set only. Note that the performance of subensembles that are optimal in training need not be optimal in terms of the generalization error. Fig. 3 displays the evolution of the average train and test errors of the subensembles obtained by Ordered Aggregation, SDP-pruning and exhaustive search as a function of the subensemble size. The error of the original bagging ensemble is also displayed for comparison. In

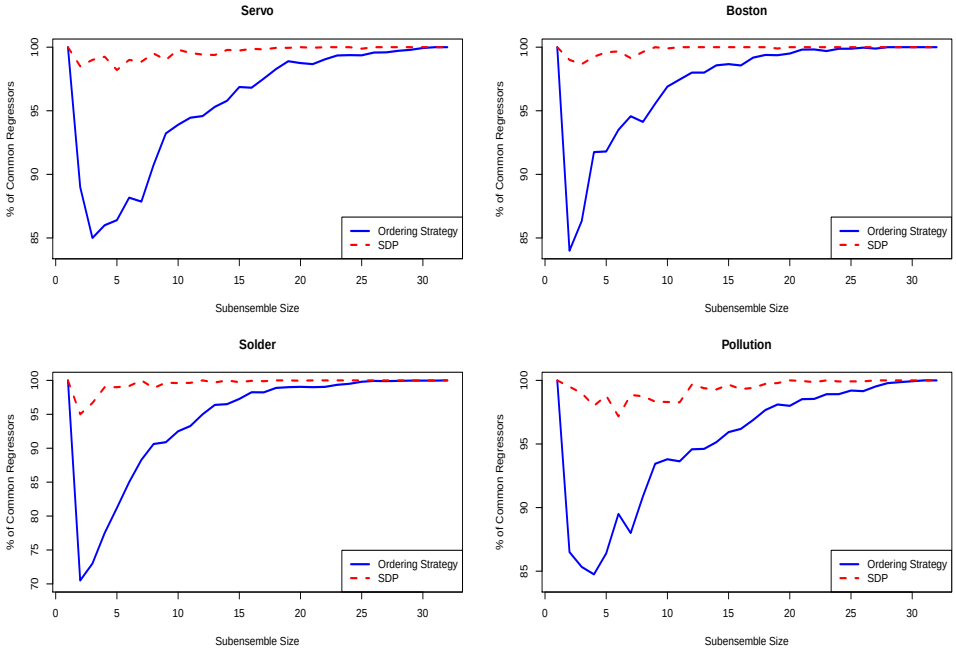

Fig. 2: Average fraction of common regressors between the optimal subensemble and subensembles obtained by SDP-pruning (top curve), or Ordered Aggregation (bottom curve) as a function of the subensemble size.

standard bagging the error decreases approximately monotonically as more regressors are incorporated into the ensemble. This behavior should be expected from the random order in the aggregation of regressors.

For the majority of the problems investigated, the empirical curves that trace the dependence of the error of the optimal and near-optimal ensembles as a function of ensemble size are qualitatively similar both for the training error and for the test. They exhibit an initial reduction of the error, which is steeper than in the standard bagging ensemble. At intermediate ensemble sizes the error curves display a fairly broad minimum. Beyond this minimum the error slowly increases and eventually approaches the error level of the complete bagging ensemble from below.

The training error curves for the subensembles obtained by exhaustive search are a lower bound to the corresponding curves of the subensembles obtained by means of SDP-pruning and Ordered Aggregation. The training error curve for SDP-pruning is almost identical to the one corresponding to

optimal pruning. The curves for Ordered Aggregation are only slightly above the optimal ones. This means that even though some regressors included by ordered aggregation replace others that appear in the optimal solution, their aggregate performance is still close to optimal.

The test error curves are less smooth than the training error curves. In a small number of regression problems, such as *Solder*, the test error curves for the ensembles identified by SDP or Ordered Aggregation do not exhibit a minimum. For such problems these pruning methods are not effective, as evidenced by the results summarized in Table 2. In the majority of the problems investigated SDP and Ordered Aggregation are effective pruning techniques. In these regression tasks, the patterns in the error curves for the training set are also present in the test error curves, albeit with some differences. Similarly as in training a minimum in the test error curves appears at intermediate subensemble sizes. The value of this minimum error for the pruned subensembles is also significantly lower than the best error of a standard bagging ensemble. In contrast to the average training error curves, there are no clear differences in generalization performance among the different subensembles. Another salient difference is that the minimum appears earlier in the aggregation process in the training error curve than in the error curve for the test set. An apparent manifestation of overfitting is that the minimum generally appears at larger subensemble sizes in the test error curves than in the training error curves. The subensembles that minimize the error on the training data are generally smaller than the subensembles that are optimal when the error is estimated on an independent test set. This means that, in general, it is difficult to estimate from the training data where exactly the global minimum in the test error curve lies. Nevertheless, the error curves are rather flat around the minimum. Therefore, small errors in the estimation of the size of the ensemble that corresponds to the minimum do not have a large impact on the generalization performance.

3.2. Bias-variance-covariance decomposition of the generalization error

A bias-variance-covariance decomposition of the generalization error is carried out to analyze the dependence of the error on the size of the ensemble and to investigate the mechanisms by which ensemble pruning improves generalization performance. Since the regressors that make up a bagging ensemble are generated from bootstrap samples of the same original training data $\mathcal{D}$ and they use the same learning algorithm (e.g. neural networks

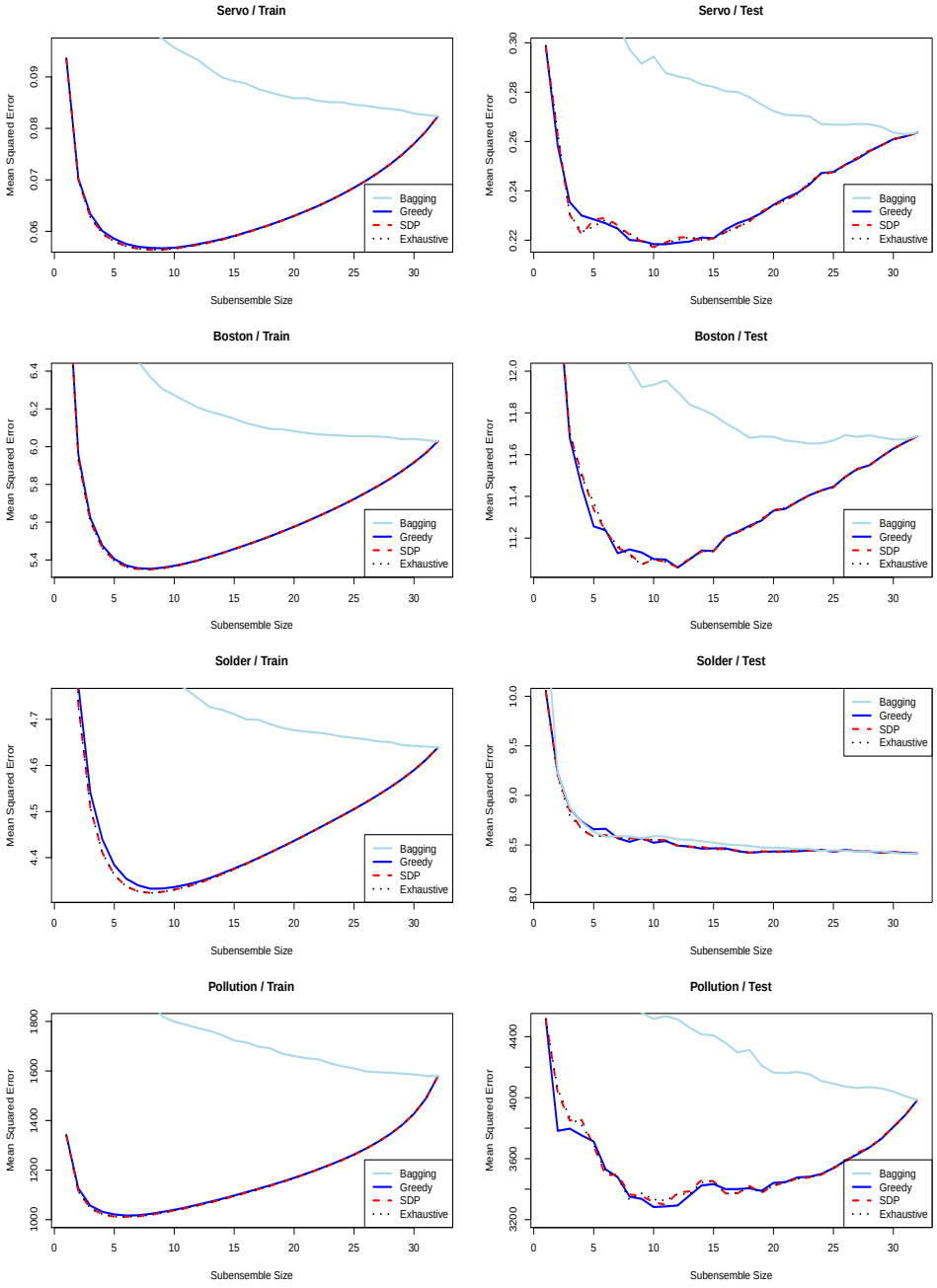

Fig. 3: Average training (left column) and test (right column) errors of the subensembles obtained by means Ordered Aggregation, SDP-pruning and the optimal subensembles found by the exhaustive search algorithm. Optimality is defined in terms of the error in the training set.

with a fixed architecture), they can be seen as realizations of a random variable drawn from a probability distribution $\mathcal{P}\left(\hat{f}_i(\cdot)\right)$. As shown in [49] the squared error of a regression ensemble of size u for a test instance $(\mathbf{x}, y)$ is composed of three terms: the average bias, the average variance and the average covariance of the individual regressors in the ensemble

$$\mathcal{L}^{(u)}(\mathbf{x}, y) = u^{-1}\overline{\mathrm{Var}}(\mathbf{x}) + (1 - u^{-1})\overline{\mathrm{Cov}}(\mathbf{x}) + \overline{\mathrm{Bias}}^2(\mathbf{x}, y), \tag{16}$$

with the definitions

$$\begin{aligned} \overline{\mathrm{Bias}}(\mathbf{x}, y) = u^{-1}\sum_{i=1}^{u}\mathrm{Bias}(\hat{f}_i|\mathbf{x}, y), \qquad \overline{\mathrm{Var}}(\mathbf{x}) = u^{-1}\sum_{i=1}^{u}\mathrm{Var}(\hat{f}_i|\mathbf{x}), \\ \overline{\mathrm{Cov}}(\mathbf{x}) = (u-1)^{-1}u^{-1}\sum_{j\neq i}\mathrm{Cov}(\hat{f}_i, \hat{f}_j|\mathbf{x})\,, \end{aligned} \tag{17}$$

where

$$\mathrm{Bias}(\hat{f}_i|\mathbf{x}, y) = \mathbb{E}_{\mathcal{D}}\left\{\hat{f}_i(\mathbf{x}) - y\right\}, \tag{18}$$

$$\mathrm{Var}(\hat{f}_i|\mathbf{x}) = \mathbb{E}_{\mathcal{D}}\left\{\left(\hat{f}_i(\mathbf{x}) - \mathbb{E}_{\mathcal{D}}\left\{\hat{f}_i(\mathbf{x})\right\}\right)^2\right\}, \tag{19}$$

$$\mathrm{Cov}(\hat{f}_i, \hat{f}_j|\mathbf{x}) = \mathbb{E}_{\mathcal{D}}\left\{\left(\hat{f}_i(\mathbf{x}) - \mathbb{E}_{\mathcal{D}}\left\{\hat{f}_i(\mathbf{x})\right\}\right)\left(\hat{f}_j(\mathbf{x}) - \mathbb{E}_{\mathcal{D}}\left\{\hat{f}_j(\mathbf{x})\right\}\right)\right\} \tag{20}$$

are the bias, variance and covariance of the ith regressor in the ensemble. In standard bagging the variances and biases of all regressors are equal. Therefore, the expected regression error of a standard bagging ensemble of size u is

$$\mathcal{L}^{(u)}(\mathbf{x}, y) = u^{-1}\mathrm{Var}(\mathbf{x}) + (1 - u^{-1})\overline{\mathrm{Cov}}(\mathbf{x}) + \mathrm{Bias}^2(\mathbf{x}, y)\,, \tag{21}$$

where $\mathrm{Var}(\mathbf{x})$ and $\mathrm{Bias}(\mathbf{x}, y)$ are the expected variance and bias of a regressor drawn from the distribution $\mathcal{P}\left(\hat{f}_i(\cdot)\right)$ and $\overline{\mathrm{Cov}}(\mathbf{x})$ is their average covariance.

Consider the original bagging ensemble of size M. The selection strategies used in SDP-pruning and in aggregation ordering have the effect of modifying the distribution of the regressors that make up the near-optimal subensembles of size $u \leq M$. Instead of being independent of the subensemble size, this distribution changes as new regressors are aggregated into the partial subensemble. Let $\mathcal{P}\left(\hat{f}_i(\cdot); u\right)$ denote the distribution of the regressors

that are part of the near-optimal subensemble of size u. The bias-variance-covariance error decomposition for the near-optimal subensembles is

$$\mathcal{L}^{(u)}(\mathbf{x}, y) = u^{-1}\mathrm{Var}_u(\mathbf{x}) + (1 - u^{-1})\overline{\mathrm{Cov}}_u(\mathbf{x}) + \mathrm{Bias}_u^2(\mathbf{x}, y) \tag{22}$$

where $\mathrm{Var}_u(\mathbf{x}, y)$ and $\mathrm{Bias}_u(\mathbf{x}, y)$ are the expected variance and bias of a regressor drawn from the distribution $\mathcal{P}\left(\hat{f}_i(\cdot); u\right)$ and $\overline{\mathrm{Cov}}_u(\mathbf{x})$ is the average covariance between the members of a near-optimal subensemble of size u. As a result of the selection strategies, the values of $\mathrm{Bias}_u(\mathbf{x}, y)$ and $\mathrm{Var}_u(\mathbf{x})$ (the average bias and variance of a regressor in a near-optimal subensemble of size u) are expected to be lower than $\mathrm{Bias}(\mathbf{x})$ and $\mathrm{Var}(\mathbf{x})$ (the average bias and variance bias and variance of a regressor in a standard bagging ensemble) at least up to intermediate subensemble sizes. A similar behavior is expected for $\overline{\mathrm{Cov}}_u(\mathbf{x})$. In this manner, near-optimal subensembles can achieve a lower error than standard bagging subensembles for $u = 1, 2, \ldots, (M-1)$. Note that when $u = M$, where M is the size of the original ensemble, $\mathcal{P}\left(\hat{f}_i(\cdot); u\right) = \mathcal{P}\left(\hat{f}_i(\cdot)\right)$, $\overline{\mathrm{Cov}}_M(\mathbf{x}) = \overline{\mathrm{Cov}}(\mathbf{x})$ and the errors of all ensembles are equal.

The asymptotic error of bagging for such a test instance is

$$\lim_{u\to\infty}\mathcal{L}^{(u)}(\mathbf{x}, y) = \mathrm{Bias}^2(\mathbf{x}, y) + \overline{\mathrm{Cov}}(\mathbf{x}) \geq 0. \tag{23}$$

Hence, it is possible that the subensemble at iteration u has a lower error than this asymptotic limit if the inequality

$$u^{-1}\mathrm{Var}_u(\mathbf{x}) + (1 - u^{-1})\overline{\mathrm{Cov}}_u(\mathbf{x}) + \mathrm{Bias}_u^2(\mathbf{x}, y) < \mathrm{Bias}^2(\mathbf{x}, y) + \overline{\mathrm{Cov}}(\mathbf{x}) \tag{24}$$

is satisfied. This inequality is fulfilled if the approximate strategy selects from the complete ensemble a set of regressors which have low bias, low variance and also small or negative correlations. In the experiments carried out the inequality is fulfilled for a large range of subensemble sizes.

Fig. 4 and Fig. 5 display the average value over the test instances of the squared error, and of its components (squared bias, variance and covariance), as a function of ensemble size for standard bagging, Ordered Aggregation and SDP-pruning for the synthetic regression problems *Peak* and *Friedman1* and for the real-world problems *Pollution* and *Chick*. The characteristics of these problems are displayed in Table 1. In each problem the original bagging ensemble is composed of 100 neural network with 5 units in the hidden layer. These networks are generated on independent bootstrap samples from a fixed

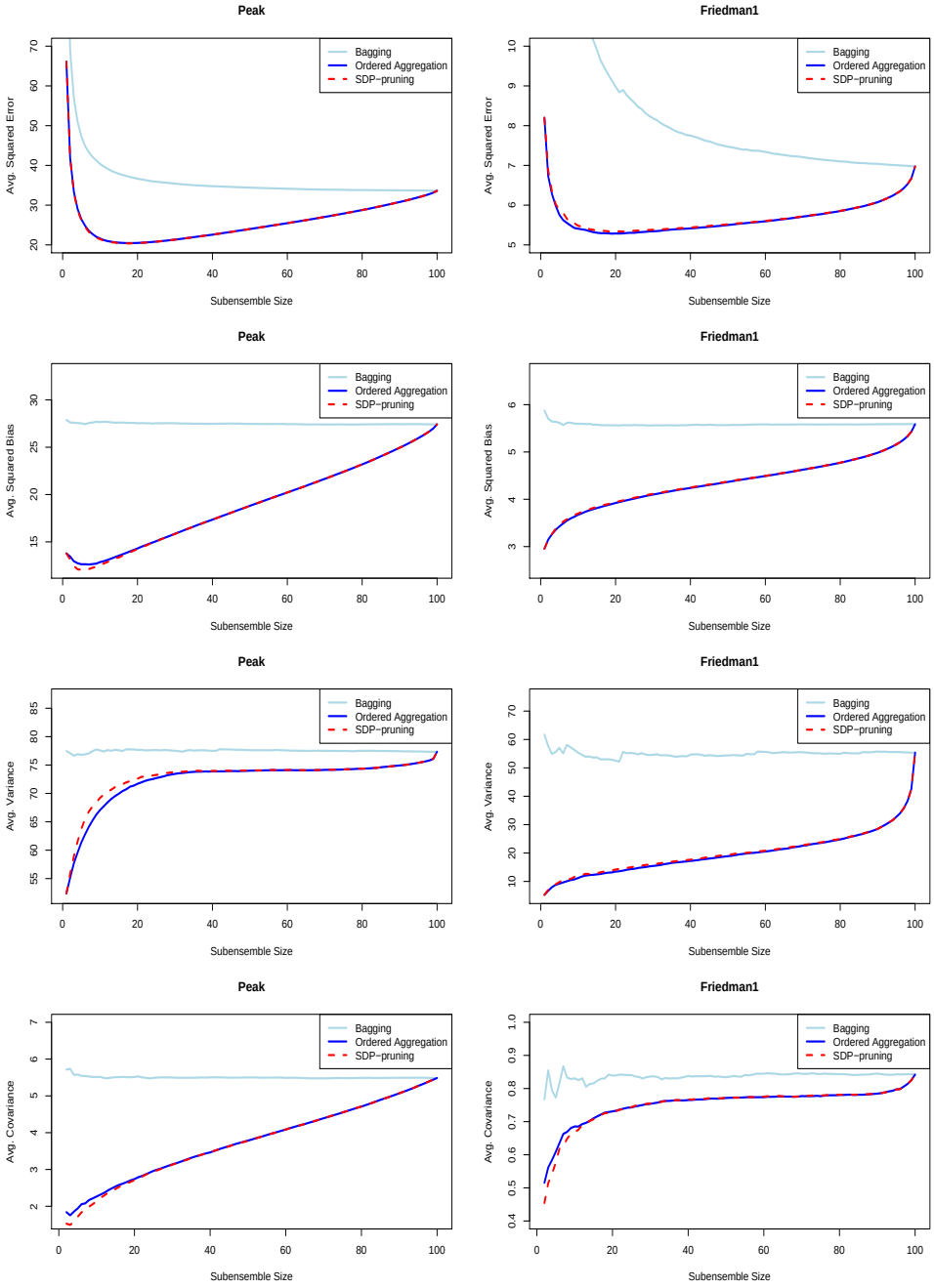

Fig. 4: From top to bottom: average over the test instances of the squared error, squared bias, variance and covariance as a function of ensemble size for bagging, Ordered Aggregation and SDP-pruning for the synthetic regression problems *Peak* and *Friedman1*.

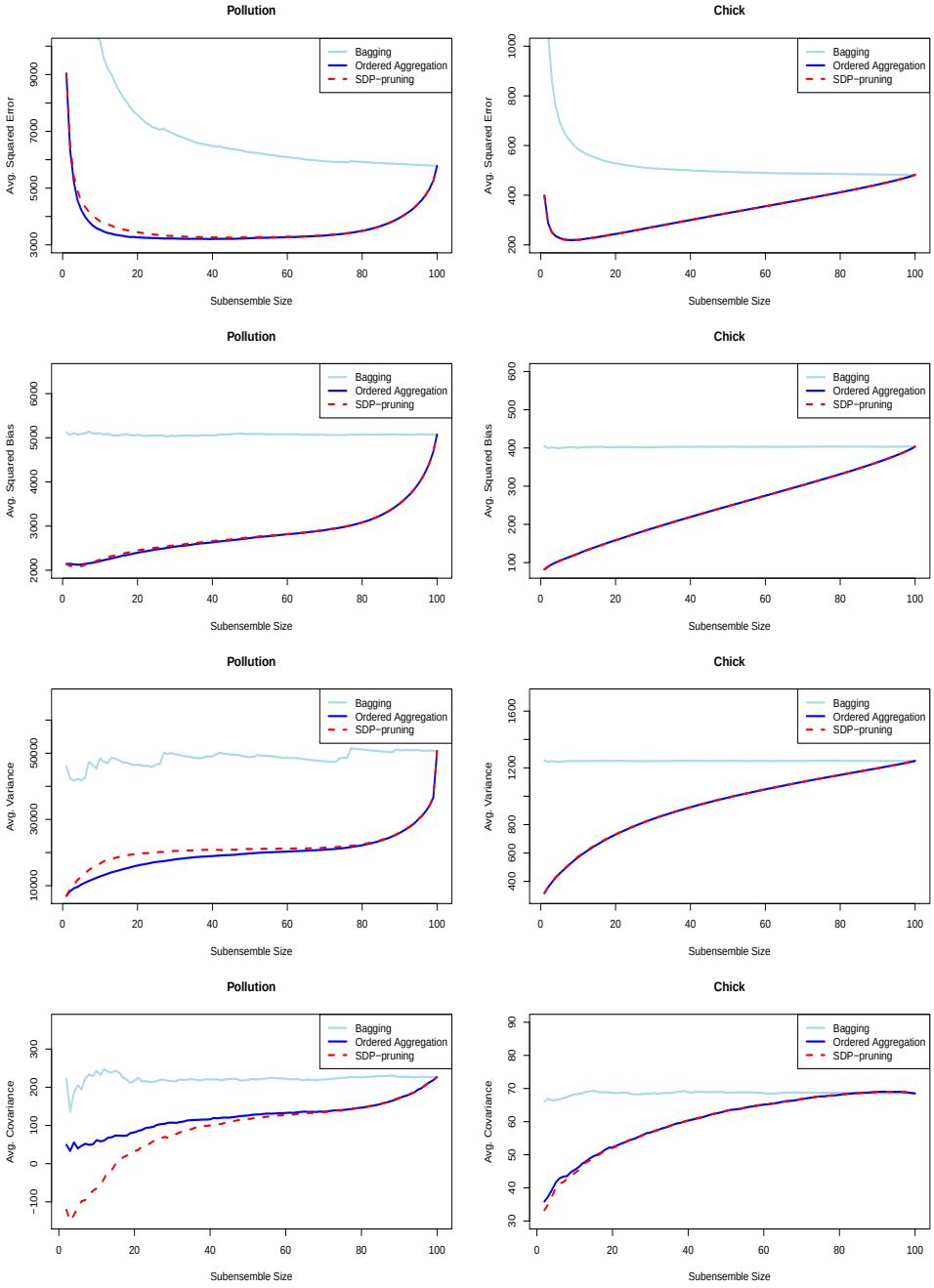

Fig. 5: From top to bottom: average over the test instances of the squared error, squared bias, variance and covariance as a function of ensemble size for bagging, Ordered Aggregation and SDP-pruning for the regression problems *Pollution* and *Chick*.

training set. In the case of the synthetic problems *Peak* and *Friedman1* the training set is randomly generated and contains 200 instances. The test set is independent of the training data and consists of 2000 elements. In the case of the real-world problems *Pollution* and *Chick*, sub-sampling is used to carry out the estimations [40]. Specifically, the data are randomly split in a training set and a test set containing 2/3 and 1/3 of the instances, respectively. The values of the test error and of its components are estimated by averaging over 1000 realizations of each problem (for *Pollution* 5000 realizations are used as a consequence of the reduced number of instances available for this problem). The error curves exhibit similar features to those in Fig. 3. For standard bagging, the values of the average squared bias variance and covariance remain approximately constant as the number of regressors in the ensemble varies. In contrast with standard bagging, the average squared bias, variance and covariance typically increase with the size of the subensembles selected using Ordered Aggregation and SDP-pruning. A lower covariance among the ensemble members was also observed in [29] and is a consequence of minimizing (5) [12, 29].

According to (21) and (22), as the size of the ensemble grows, the squared bias and the covariance become the most important terms in the error decomposition. The variance term is inversely proportional to u, the size of the subensemble. Therefore, its contribution to the generalization error becomes smaller as the number of predictors of the ensemble increases. For sufficiently large subensemble sizes the test error of the complete ensemble in standard bagging is approximately equal to the sum of the squared bias and the covariance term (23). In consequence, even though the predictors included in the smaller subensembles by SDP-pruning and Ordered Aggregation also have low variance, the main reason for the improvements in performance is that the first regressors incorporated by these strategies have low bias and also small or negative covariance.

3.3. Generalization Performance

In this section we assess the generalization performance of SDP-pruning and Ordered Aggregation and compare these techniques with predictors of different types: a single neural network, bagging, Adaboost.R2, negative correlation learning ensembles (NCL) and regularized stacked generalization combined the lasso (SG-lasso), as described in [43], and the ensemble pruning methods of Perrone and Cooper [39], and GASEN[56]. The experiments are carried out on 24 regression problems from the UCI-Repository [4], from the

Weka Data Mining Tool [52] and from other sources [10, 14, 20, 28, 42]. They include synthetic and real-world problems from different fields of application. Table 1 displays the number of instances, the number of attributes and the source of the different datasets considered.

For each real-world dataset, 10×10-fold cross-validation is used to estimate the squared regression error. For the synthetic datasets (*Friedman1, Friedman2, Friedman3*, and *Peak*) the values reported are averages over 100 independent realizations of the train and test datasets. The training set and the test set contain 200 and 2000 instances, respectively. Data attributes are normalized so that they have zero mean and unit variance on the training set for each dataset. The computation of the estimates for each training and testing partition involves the following steps: (i) Generate a bagging ensemble of 100 neural networks from the training set using bootstrap sampling. The architecture of these neural networks and the weight decay constant used is determined as in Section 3.1. Then, the neural networks are trained over 1000 epochs using the best combination of parameters found. (ii) A single neural network and a boosting ensemble of size 100 are also built using the same network configuration as in the bagging ensemble. The boosting ensemble is generated using the Adaboost.R2 algorithm with a linear loss function [18]. A 100 neural network NCL ensemble is generated as described in [29] by training each network during 100 epochs using gradient descent. The weights of the networks are initially set to the values provided by bagging. The λ parameter of the NCL ensemble is set equal to 0.99, as suggested in [12]. Instead of using a fixed value, λ could be determined by cross-validation. However, the results obtained are very similar and do not compensate the increased computational cost. The resulting predictors are then used as a benchmark for performance comparison. (iii) A subensemble of 20 neural networks (i.e. $u = 20$) is extracted from the original pool of classifiers generated by bagging using SDP-pruning, as described in Section 2.1. The complete bagging ensemble is then ordered according to Ordered Aggregation and the first 20 regressors of the ordered sequence are selected and aggregated in a pruned subensemble. Additionally, the bagging ensemble is pruned using the technique described by Perrone and Cooper in [39] and using the genetic algorithm GASEN introduced in [56]. GASEN is configured to select subensembles of the same size as SDP-pruning and Ordered Aggregation (20 neural networks from the original pool of 100). These pruned predictors are also used as a benchmark for performance comparison. (iv) We also generate a regularized linear stacked generalization ensemble (SG-lasso), where

the coefficients that combine the outputs of the different predictors are estimated using the lasso, as described in [43]. The initial predictors used in this ensemble method are the 100 neural networks generated in bagging. The regularization parameter of the lasso is selected from a grid of 100 potential values using a nested 10-fold cross-validation to estimate the mean squared error. Only the training data are used to compute this estimate. (v) The error of each method is estimated on the corresponding test set.

Table 2 shows the average mean squared error estimated either by 10×10-fold cross-validation (real-world problems) or in the test set (synthetic data) for each dataset and each prediction method. The figures displayed are scaled by a factor shown in the first column of the table. The next five columns of the table display the error of a single neural network, the complete bagging ensemble, the boosting ensemble, the NCL ensemble and the regularized linear stacked generalization ensemble (SG-lasso), respectively. Finally, the four last columns of the table display the average errors of the bagging subensembles selected by SDP-pruning, Ordered Aggregation, the method of Perrone and Cooper and the genetic algorithm GASEN. To determine whether the observed improvements in accuracy are statistically significant a paired *Wilcoxon-test* [51] is performed, as suggested in [16]. Error values that are significantly better than bagging according to the Wilcoxon text at an α-value of 0.05 are highlighted in boldface. Error values that are significantly worse than bagging are underlined. Similarly, error values that are significantly better than boosting are marked with the symbol $\blacktriangleleft$. Values that are significantly worse than boosting are marked with the symbol $\triangleleft$.

From these results one can see that bagging ensembles often outperform a single neural network. NCL ensembles and regularized linear stacked generalization ensembles also provide more accurate predictions than bagging or boosting in most of the problems investigated. The subensembles obtained by SDP-pruning and Ordered Aggregation that contain only 20 regressors have a good overall generalization performance in the regression problems investigated. In general, these pruned bagging ensembles outperform single networks, boosting ensembles and complete bagging ensembles. Table 3 shows that the pruning rates obtained by SDP-pruning and Ordered Aggregation (20%) are generally higher than those achieved by the method proposed by Perrone and Cooper ($\approx$ 33 networks on average). Regularized linear stacked generalization ensembles contain an average of $\approx$ 19 neural networks.

An overall comparison of the performance of the different methods in

Table 2: Average mean squared error normalized by the corresponding scaling factor. The figures displayed correspond to a single neural network, complete bagging ensembles, Adaboost.R2 ensembles, negative correlation learning (NCL) ensembles, regularized linear stacked generalization ensembles (SG-lasso), and pruned bagging ensembles selected by SDP-pruning, Ordered Aggregation (OA), the method of Perrone and Cooper (P&C) and the genetic algorithm GASEN.

Dataset	**Scaling Factor**	**Neural Net**	**Bagging**	**Adaboost**	**NCL**	**SG-lasso**	**Pruned Bagging**			
							SDP	**OA**	**P&C**	**GASEN**
Attitude	$\times 10^{1}$	9.45±7.90	8.63±4.95	8.83±4.97	8.59±5.51	8.54±5.92	**8.29±5.18** ◀	**8.29±5.11** ◀	**8.35±5.55** ◀	**8.33±5.01** ◀
AutoPrice	$\times 10^{7}$	1.17±0.86 ◁	0.74±0.59 ◁	**0.56±0.40**	**0.71±0.58** ◁	**0.62±0.47** ◁	**0.63±0.50** ◁	**0.63±0.50** ◁	**0.65±0.50** ◁	**0.68±0.51** ◁
Bodyfat	$\times 1$	2.96±3.76	1.89±2.76 ◀	2.55±2.00	2.33±2.66 ◀	2.22±2.74 ◀	2.12±2.78 ◀	2.10±2.78 ◀	2.15±2.87 ◀	2.01±2.81 ◀
Bolts	$\times 10^{1}$	**5.31±7.70** ◀	6.56±7.69 ◀	9.72±13.28	**6.07±7.80** ◀	**4.01±6.26** ◀	**4.57±7.17** ◀	**4.50±7.01** ◀	**4.43±7.21** ◀	**5.81±7.44** ◀
Boston	$\times 10^{1}$	1.27±0.64 ◁	1.15±0.64 ◁	**1.00±0.43**	**1.02±0.57**	**1.06±0.52**	**1.07±0.55** ◁	**1.07±0.55** ◁	**1.10±0.57** ◁	1.13±0.60 ◁
Chick	$\times 10^{1}$	**5.44±3.66** ◁	6.57±3.58 ◁	**3.89±2.74**	**3.93±2.40**	**4.62±2.39** ◁	**4.82±2.43** ◁	**4.88±2.60** ◁	**5.77±3.40** ◁	**6.23±3.58** ◁
Concrete Slump	$\times 10^{1}$	3.41±2.15	3.23±1.54	3.12±1.43	**2.98±1.45** ◀	**2.81±1.40** ◀	**2.85±1.42** ◀	**2.85±1.44** ◀	**2.97±1.48** ◀	3.16±1.60
Fires	$\times 10^{3}$	1.82±4.04	1.55±3.49 ◁	**1.15±2.59**	**1.48±3.75**	1.62±3.43 ◁	1.32±3.06 ◁	1.31±3.05 ◁	1.58±3.51 ◁	1.63±3.40 ◁
Friedman1	$\times 1$	**4.82±1.29** ◀	4.85±0.44 ◀	5.05±0.76	**4.34±0.46** ◀	**4.23±0.53** ◀	**4.45±0.46** ◀	**4.45±0.46** ◀	**4.55±0.48** ◀	**4.71±0.45** ◀
Friedman2	$\times 10^{4}$	2.68±0.89 ◁	2.19±0.16	2.17±0.13	2.20±0.16	**2.02±0.14** ◀	**2.06±0.14** ◀	**2.06±0.15** ◀	**2.06±0.15** ◀	**2.15±0.15** ◀
Friedman3	$\times 10^{-2}$	1.83±0.25 ◁	1.70±0.21 ◀	1.74±0.22	**1.69±0.21** ◀	1.69±0.22 ◀	**1.67±0.22** ◀	**1.67±0.22** ◀	**1.68±0.22** ◀	1.71±0.22 ◀
Loblolly	$\times 1$	**4.45±8.93** ◀	6.39±6.36 ◀	10.93±9.60	**3.95±6.30** ◀	**4.27±5.65** ◀	**3.96±6.24** ◀	**3.93±6.22** ◀	**3.96±5.91** ◀	**6.07±8.12** ◀
Longley	$\times 10^{-1}$	**8.25±16.10** ◁	13.08±24.85 ◁	**5.63±8.13**	**5.21±5.66**	**5.14±9.54**	**4.79±7.82** ◀	**4.81±7.89** ◀	**5.20±8.94**	**10.53±45.79**
Orange	$\times 10^{2}$	2.21±1.80	1.85±1.32	1.97±1.44	**1.58±0.83** ◀	1.75±1.12	1.68±1.09 ◀	**1.66±1.06** ◀	**1.64±1.09** ◀	**1.70±1.16** ◀
Ozone	$\times 10^{1}$	1.69±0.44 ◀	1.65±0.43 ◀	1.86±0.48	1.65±0.43 ◀	1.67±0.46 ◀	1.65±0.44 ◀	1.65±0.44 ◀	1.64±0.43 ◀	1.66±0.44 ◀
Peak	$\times 10^{1}$	3.16±0.53 ◁	2.63±0.34 ◁	**2.47±0.23**	**1.98±0.28** ◀	**1.56±0.23** ◀	**2.37±0.34** ◀	**2.37±0.34** ◀	**2.59±0.33** ◁	**2.55±0.34** ◁
Pollution	$\times 10^{3}$	5.90±6.54	3.94±3.25	7.50±11.19	**2.85±1.71** ◀	4.55±9.95 ◀	**3.25±2.49** ◀	**3.20±2.47** ◀	**3.36±4.18** ◀	**3.47±2.29** ◀
Rock	$\times 10^{4}$	8.46±10.24 ◁	6.77±6.81 ◀	7.44±7.54	6.82±6.74 ◀	7.98±7.83	7.28±7.41	7.25±7.36	7.63±7.56	7.07±7.18 ◀
Sensory	$\times 10^{-1}$	5.34±0.95 ◀	5.35±0.92 ◀	5.63±0.98	**5.28±0.90** ◀	5.47±0.91 ◀	**5.19±0.91** ◀	**5.20±0.92** ◀	**5.28±0.92** ◀	5.35±0.96 ◀
Servo	$\times 10^{-1}$	3.27±3.28 ◁	2.63±2.35 ◁	**1.70±1.78**	**2.40±2.11** ◁	**1.94±1.84** ◁	**2.03±1.77** ◁	**2.05±1.81** ◁	**2.06±1.69** ◁	2.43±2.03 ◁
Solder	$\times 1$	9.04±3.37 ◀	8.36±3.13 ◀	9.44±3.53	8.83±3.37 ◀	8.48±3.33 ◀	8.43±3.23 ◀	8.41±3.20 ◀	8.33±3.12 ◀	8.41±3.14 ◀
Theoph	$\times 1$	5.30±3.77 ◁	4.43±2.27 ◁	**3.04±2.15**	**3.74±2.24** ◁	**2.87±1.75**	**3.42±2.00** ◁	**3.41±2.01** ◁	**3.25±1.93** ◁	**4.11±2.25** ◁
Tooth	$\times 10^{1}$	**1.40±0.69** ◀	1.43±0.71 ◀	1.52±0.79	**1.42±0.69** ◀	1.45±0.71 ◀	1.44±0.72 ◀	1.44±0.72 ◀	1.44±0.72 ◀	1.46±0.73
Wisconsin	$\times 10^{2}$	9.64±2.66	9.18±2.48 ◀	9.70±2.31	9.30±2.49 ◀	9.84±2.62	9.13±2.39 ◀	9.13±2.40 ◀	**9.09±2.40** ◀	9.24±2.48 ◀

Table 3: Average number of networks in the pruned subensembles obtained by the method of Perrone and Cooper (P&C) and in the ensembles generated by regularized linear stacked generalization using the lasso (SG-lasso).

Dataset	**# of Neural Networks**	
	P & C	**SG-lasso**
Attitude	16.62±12.94	10.27±2.80
AutoPrice	24.16±7.72	31.43±15.38
Bodyfat	26.87±10.03	20.35±4.24
Bolts	17.20±9.10	12.75±2.75
Boston	52.21±17.35	20.71±8.47
Chick	71.81±10.54	34.78±5.19
Concrete Slump	33.41±14.82	16.97±4.14
Fires	3.19±2.70	11.22±9.12
Friedman1	42.08±13.76	20.80±10.78
Friedman2	23.66±9.82	12.73±2.77
Friedman3	35.52±11.37	15.74±7.00
Loblolly	38.65±18.77	20.19±4.08
Longley	6.80±4.98	8.07±3.22
Orange	23.67±12.34	11.25±3.07
Ozone	35.80±18.16	14.48±5.17
Peak	90.59±6.53	53.71±5.21
Pollution	21.29±17.64	14.79±5.46
Rock	12.56±7.97	8.31±2.64
Sensory	63.64±13.72	28.64±6.47
Servo	36.43±12.82	21.85±10.94
Solder	53.76±11.09	18.62±4.41
Theoph	12.76±14.30	14.77±8.94
Tooth	15.65±8.38	10.05±3.45
Wisconsin	43.72±17.60	19.27±4.78
Average	33.42±21.33	18.82±10.17

the collection of problems investigated can be made using the framework proposed by Demšar in [16]. On the basis of the results of Friedman test on the average ranks of each algorithm in the problems investigated, the hypothesis that their performances are equivalent can be rejected at $\alpha = 0.05$. A Nemenyi test is used to determine whether the differences in average rank among the different algorithms are significant. Fig. 6 displays the results of this test for $\alpha = 0.05$. In this figure, algorithms whose differences in performance are not statistically significant are linked with a solid black line. Differences in performance between algorithms whose average ranks are further than a critical distance (CD) are statistically significant.

In the collection of regression problems investigated pruned ensembles, regularized linear stacked generalization ensembles and negative correlation learning ensembles have a better overall performance than the corresponding complete bagging ensembles, boosting ensembles and single neural networks. Notwithstanding, the differences in average rank are only statistically significant for SDP-pruning and Ordered Aggregation. In these problems subensembles selected by Ordered Aggregation or SDP-pruning have better overall generalization performance than the subensembles selected by the method of Perrone and Cooper, SG-lasso and the genetic algorithm GASEN. Furthermore, the differences in the average ranks with respect to this latter method are statistically significant. Ordered Aggregation also performs slightly better than SDP-pruning, although the difference between their average ranks is not statistically significant.

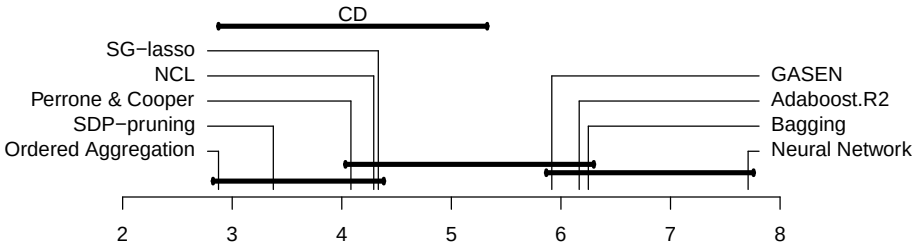

Fig. 6: Results of a Nemenyi test on the average ranks of the different regression systems: a single neural network, bagging, boosting (Adaboost.R2), negative correlation learning (NCL), regularized linear stacked generalization (SG-lasso) and pruned subensembles selected by SDP-pruning, Ordered Aggregation, the method of Perrone and Cooper and GASEN. The critical difference (CD) between average ranks is displayed at the top of the figure.

The curves in Fig. 7 trace the dependence of the average squared error

for bagging, Ordered Aggregation, boosting, negative correlation learning (NCL) and regularized linear stacked generalization (SG-lasso) as a function of the ensemble size for a representative subset of the regression problems investigated. For SDP-pruning, because of its high computational cost, only the average error of the subensemble of size $u = 20$ is displayed. For the genetic algorithm GASEN, the method of Perrone and Cooper and regularized linear stacked generalization we display their corresponding mean squared error evaluated at the average number of networks used for prediction. The average error of a single neural network is also marked as a horizontal line for reference. The curves for bagging and Ordered Aggregation exhibit a qualitative behavior that is similar to the curves depicted in Fig. 3. The error decreases almost monotonically as a function of the size of a randomly ordered bagging ensemble. Modifying the order of the aggregation according to the procedure described leads to an initially steeper descent of the error curves, which, with some exceptions (e.g. *Solder*, *Tooth*) is followed by a broad minimum and a final rise to the error level of the complete bagging ensemble. In fact, because the error curves are rather flat around the minimum, any value between 15% and 25% percent of the original pool of regressors yields similar results. The performance of the subensembles selected by SDP-pruning is similar to the performance of the subensembles obtained by Ordered Aggregation for size $u = 20$. However, in some cases SDP-pruning leads to slightly higher error rates (e.g. *Tooth*). The subensembles selected by GASEN exhibit in general a higher prediction error than the subensembles selected by either SDP-pruning or Ordered Aggregation. The method of Perrone and Cooper provides better results than those of these two pruning strategies in some problems (e.g. *Tooth*, *Wisconsin*). Nevertheless, in several other problems worse results are observed (e.g. *Pollution*, *Peak*, *Friedman1*, *Boston*). In particular, the method of Perrone and Cooper tends to underprune and selects subensembles that are too large. Regularized linear stacked generalization (SG-lasso) generates the most accurate ensembles in some problems, e.g. *Peak* and *Friedman1*. Nevertheless, the prediction performance of this ensemble method is severely impaired in some cases, e.g. *Wisconsin* and *Pollution*. Finally, NCL provides in general better results than bagging and boosting with a few exceptions (e.g. *Solder*, *Wisconsin*). This is in agreement with the findings of [12].

The behavior of boosting ensembles is more erratic, reflecting the lack of robustness of this method. Boosting outperforms bagging, Ordered Aggregation and SDP-pruning in a few regression problems (e.g. *Boston*, *Peak*).

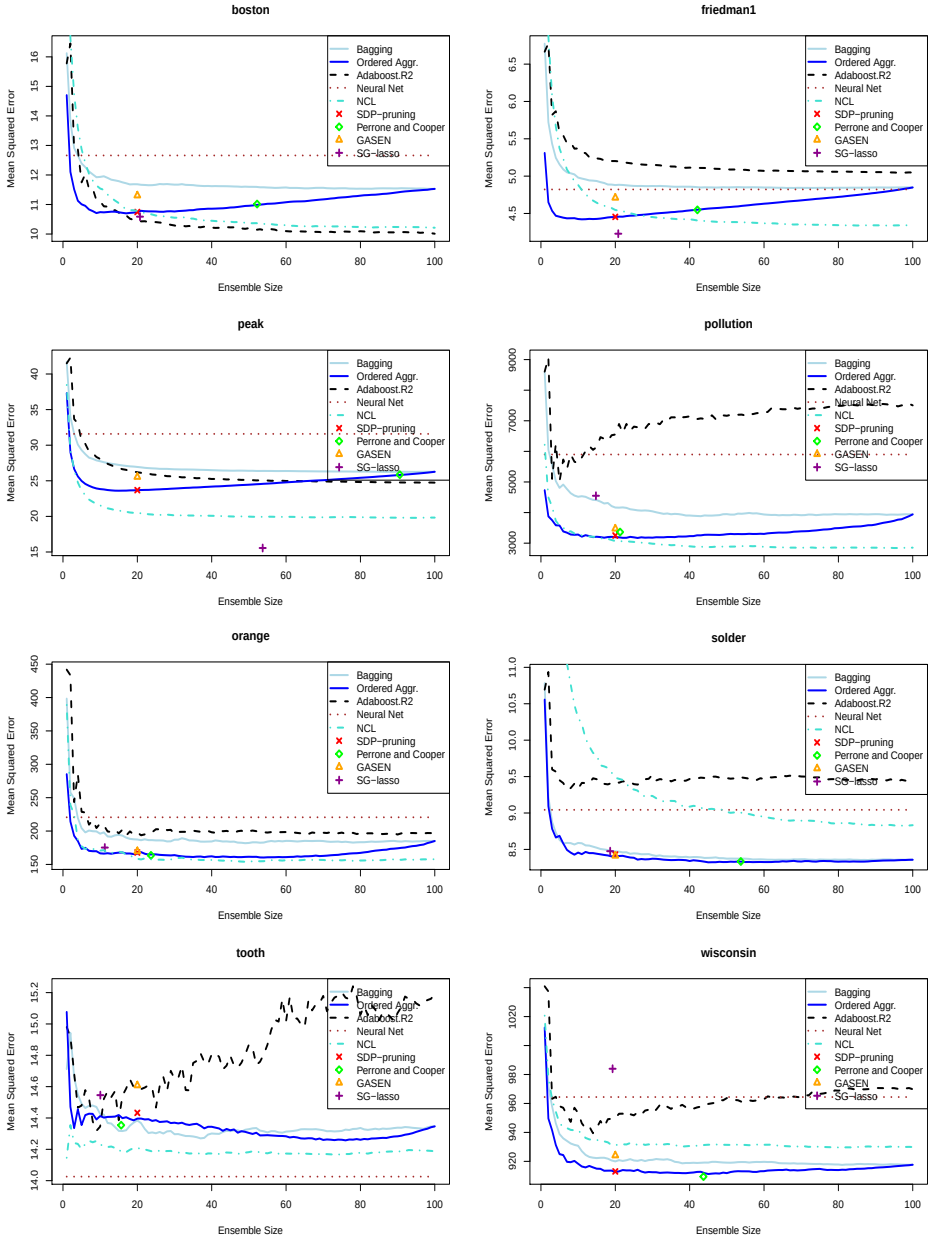

Fig. 7: Error curves for standard bagging, Ordered Aggregation, negative correlation learning (NCL) and Adaboost.R2 as a function of the ensemble size for a variety of regression problems. The error of a single Neural Network is displayed as an horizontal line for reference. The error for a subensemble of size $u = 20$ selected with SDP-pruning is marked with a cross. The average error of the method of Perrone and Cooper, the genetic algorithm GASEN and regularized linear stacked generalization (SG-lasso) are also displayed at the corresponding average number of selected networks.

However, boosting has a detrimental effect in others (e.g. *Tooth, Pollution, Wisconsin*). A similar deterioration of performance of boosting has been detected in noisy classification problems [17, 35]. The origin of this behavior is that boosting tends to assign larger weights to incorrectly labeled training instances. The learning algorithm is thus forced to reduce the error rate of those instances. This emphasis on examples that are difficult to predict tends to distort the original problem, leading to overfitting and consequently, to a poor generalization performance.

4. Conclusions

Extracting an optimal subensemble from an original regression ensemble is a difficult problem that can be shown to be NP-hard. In this paper we analyze two approximate techniques to address this problem. The first one is based on a Semidefinite Programming relaxation of the original problem. This method is an extension to regression ensembles of the work [55], which introduced SDP-pruning for classification ensembles. The second technique is Ordered Aggregation[26]. This is a forward selection strategy starts with an empty subensemble and incorporates in each step the regressor that reduces the training error of the current subensemble the most. Even though these approximate strategies can lead to suboptimal solutions, a detailed analysis in ensembles of intermediate size shows that the subsets selected have a near-optimal performance and share a large percentage of regressors with optimal subensembles obtained by exhaustive search.

The error of standard bagging ensembles typically decreases monotonically as the size of the ensemble is increased. By contrast, for the subsets selected by these techniques, the curves that trace the dependence of the subensemble error as a function of its size show that the minimum error is attained in subensembles of intermediate size. The features of these curves are qualitatively similar for both the training and testing errors. However, the minimum in the test error curves appears at larger sizes than in the curves for the training error. This minimum is generally below the asymptotic error of the complete bagging ensemble. This means that generalization performance of the ensemble can be improved by retaining only a subset of the regressors generated. A bias-variance-covariance decomposition of the test error shows that the key to the improvements in generalization performance is the selection of subsets of regressors whose bias is low and whose correlations are small or negative.

An extensive empirical investigation shows that pruned ensembles obtained by retaining only 20% of the original networks in a bagging ensemble, either using Ordered Aggregation or SDP-pruning, have the best overall performance. In most of the regression problems analyzed the generalization accuracy of Ordered Aggregation is slightly better than SDP. However, the differences found are not statistically significant. By contrast, the improvements over a single neural network, complete bagging ensembles, ensembles generated with Adaboost.R2 and pruned bagging ensembles identified by the GASEN algorithm [56] are statistically significant. Ensembles generated by negative correlation learning (NCL) [29], regularized stacked generalization (SG-lasso) [43], and pruned subensembles obtained using the method proposed by Perrone and Cooper in [39] or with the GASEN algorithm [56] have a slightly better overall performance than bagging or boosting, but the differences in rank are not statistically significant. The method of Perrone and Cooper has the drawback that the pruning rate is not fixed beforehand. Different pruning rates are achieved in the different regression problems. In general, they are larger than the 20% rate selected for Ordered Aggregation, SDP-pruning and GASEN. Regularized linear stacked generalization and NCL produce ensembles that often have worse prediction performance than the subensembles identified by SDP-pruning and Ordered Aggregation. However, the differences in average rank are not statically significant. The number of networks selected in stacked generalization ensembles is on average very similar to the size of the subensembles identified by these two approximate ensemble pruning strategies. NCL uses a larger number of networks for prediction.

An advantage of Ordered Aggregation is that it is a simple forward selection procedure that generates a nested sequence of subensembles of increasing size at no extra cost. By contrast, if SDP is used, one needs to solve a different semi-definite programming problem for each subensemble size. This entails a correlative increase in computational costs.

Acknowledgment

The authors acknowledge support from the Spanish Ministerio de Ciencia e Innovación, project TIN2010-21575-C02-02.

A. Optimal subensemble selection is an NP-hard problem

In this section we show that finding the subensemble with the minimum training error is an NP-hard problem. The proof proceeds by finding a problem that is known to be NP-complete and then showing that every instance of such problem can be reduced in polynomial time to the optimal subensemble selection problem [15, 22]. Consider the *Subset Sum* problem [15]. This problem consists in extracting from a given set of integers $\mathcal{S} = \{n_1, n_2 \ldots, n_M : n_i \in \mathbb{Z}\}$ a non-empty subset of elements whose sum is equal to zero. Assume that an algorithm $\mathcal{A}$ exists that can find in polynomial time the subensemble of an ensemble regressors whose combined prediction has the lowest mean squared error (MSE) on the dataset $\mathcal{D}$. If this were the case, the *Subset Sum* problem would also be solvable in polynomial time. To show how this could be accomplished, consider an ensemble of M regressors. Assume that the ith regressor in the ensemble outputs the integer value $n_i \in \mathcal{S}$ independently of the input. Define a regression problem that consists in predicting the value 0 independently of the input. The goal is then to select a non-empty subensemble $\{s_1, s_2, \ldots, s_u\}$, $1 \leq u \leq M$ whose combined prediction ($u^{-1}\sum_{i=1}^{u} n_{s_i}$, independent of the input) is as close to zero as possible. This can be achieved by minimizing the squared prediction error

$$MSE \equiv \left(u^{-1} \sum_{i=1}^{u} n_{s_i} - 0 \right)^2 = \left(u^{-1} \sum_{i=1}^{u} n_{s_i} \right)^2 . \tag{25}$$

Since the MSE is a non-negative value, if a subensemble whose mean squared error is zero exists, then algorithm $\mathcal{A}$ should find it in polynomial time. Finding a subensemble whose MSE is zero is equivalent to finding a subset $\{n_{s_1}, n_{s_2}, \ldots, n_{s_u}\} \subset \mathcal{S}$ whose elements sum to zero

$$MSE \equiv \left(u^{-1} \sum_{i=1}^{u} n_{s_i} \right)^2 = 0 \Longleftrightarrow \sum_{i=1}^{u} n_{s_i} = 0 \, . \tag{26}$$

Therefore, $\{n_{s_1}, n_{s_2}, \ldots, n_{s_u}\}$ would be a solution to the *Subset Sum* problem. Conversely, if no subset of $\mathcal{S}$ exists whose elements add up to zero, algorithm $\mathcal{A}$ would return a subset whose mean squared error is greater than zero. Since the *Subset Sum* is NP-complete, unless NP is equal to 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; i.e. it is NP-hard.

References

[1] Avnimelech, R., Intrator, N., 1999. Boosting regression estimators. Neural Computation 11˜(2), 499–520.

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

[3] Bauer, E., Kohavi, R., 1999. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. Machine Learning 36˜(1-2), 105–139.

[4] Blake, C.˜L., Merz, C.˜J., 1998. UCI repository of machine learning databases.
URL `http://www.ics.uci.edu/$\sim$mlearn/MLRepository.html`

[5] Borchers, B., 1999. CSDP : A C library for semidefinite programming. Optimization Methods and Software 11, 613–1623.

[6] Breiman, L., 1996. Bagging predictors. Machine Learning 24˜(2), 123–140.

[7] Breiman, L., 1996. Bias, variance, and arcing classifiers. Tech. Rep. 460, Statistics Department, University of California.

[8] Breiman, L., 1996. Heuristics of instability and stabilization in model selection. The Annals of Statistics 24˜(6), 2350–2383.

[9] Breiman, L., 1998. Arcing classifiers. The Annals of Statistics 26˜(3), 801–849.

[10] Breiman, L., 1999. Using adaptive bagging to debias regressions. Tech. rep., Statistics Department, University of California.

[11] Breiman, L., 2001. Random forests. Machine Learning 45˜(1), 5–32.

[12] Brown, G., Wyatt, J.˜L., Tino, P., 2005. Managing diversity in regression ensembles. Journal of Machine Learning Research 6, 1621–1650.

[13] Caruana, R., Niculescu-Mizil, A., 2004. Ensemble selection from libraries of models. In: Proceedings of the 21st International Conference on Machine Learning. ACM Press, New York, NY, USA.

[14] Chambers, J.˜M., Hastie, T.˜J., 1992. Statistical Models in S. Chapman & Hall, London.

[15] Cormen, T.˜H., Leiserson, C.˜H., Rivest, R.˜L., 1990. Introduction to Algorithms. The MIT Press.

[16] Demšar, J., 2006. Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research 7, 1–30.

[17] Dietterich, T.˜G., 2000. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. Machine Learning 40˜(2), 139–157.

[18] Drucker, H., 1997. Improving regressors using boosting techniques. In: Proc. 14th International Conference on Machine Learning. Morgan Kaufmann, pp. 107–115.

[19] Freund, Y., Schapire, R.˜E., 1995. A decision-theoretic generalization of on-line learning and an application to boosting. In: Proc. 2nd European Conference on Computational Learning Theory. pp. 23–37.

[20] Friedman, J.˜H., Mar. 1991. Multivariate adaptive regression splines. The Annals of Statistics 19˜(1), 1–67.

[21] Friedman, J.˜H., 2001. Greedy function approximation: A gradient boosting machine. The Annals of Statistics 29˜(5), 1189–1232.

[22] Garey, M.˜R., Johnson, D.˜S., 1990. Computers and Intractability; A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York, NY, USA.

[23] Goemans, M.˜X., Williamson, D.˜P., 1995. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. J. ACM 42˜(6), 1115–1145.

[24] Han, Q., Ye, Y., Zhang, J., 2000. An improved rounding method and semidefinite programming relaxation for graph partition. Tech. rep., Iowa City, Iowa 52242.

[25] Hastie, T., Tibshirani, R., Friedman, J.˜H., August 2001. The Elements of Statistical Learning. Springer.

[26] Hernández-Lobato, D., Martínez-Muñoz, G., Suárez, A., 2006. Pruning in ordered regression bagging ensembles. In: IEEE International Joint Conference on Neural Networks. pp. 1266 – 1273.

[27] Krogh, A., Hertz, J.˜A., 1992. A simple weight decay can improve generalization. In: Moody, J.˜E., Hanson, S.˜J., Lippmann, R.˜P. (Eds.), Advances in Neural Information Processing Systems. Vol.˜4. Morgan Kaufmann Publishers, Inc., pp. 950–957.

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

[29] Liu, Y., Yao, X., 1999. Ensemble learning via negative correlation. Neural Networks 12˜(10), 1399–1404.

[30] Margineantu, D.˜D., Dietterich, T.˜G., 1997. Pruning adaptive boosting. In: Proc. 14th International Conference on Machine Learning. Morgan Kaufmann, pp. 211–218.

[31] Martínez-Muñoz, G., Hernández-Lobato, D., Suárez, A., 2009. An analysis of ensemble pruining techniques based on ordered aggregation. IEEE Transactions on Pattern Analysis and Machine Intelligence 31, 245–259.

[32] Martínez-Muñoz, G., Sánchez-Martínez, A., Hernández-Lobato, D., Suárez, A., 2008. Class-switching neural network ensembles. Neurocomputing 71˜(13-15), 2521 – 2528.

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

[34] Martínez-Muñoz, G., Suárez, A., 2006. Pruning in ordered bagging ensembles. In: ICML '06: Proceedings of the 23rd international conference on Machine learning. ACM Press, New York, NY, USA, pp. 609–616.

[35] Martínez-Muñoz, G., Suárez, A., 2007. Using boosting to prune bagging ensembles. Pattern Recognition Letters 28˜(1), 156–165.

[36] Miller, A., 2002. Subset Selection in Regression. Chapman & Hall/CRC.

[37] Nanni, L., Lumini, A., 2009. Ensemble generation and feature selection for the identification of students with learning disabilities. Expert Systems with Applications 36˜(2, Part 2), 3896 – 3900.

[38] Nocedal, J., Wright, S.˜J., 1999. Numerical Optimization. Springer.

[39] Perrone, M.˜P., Cooper, L.˜N., 1993. When networks disagree: Ensemble methods for hybrid neural networks. In: Mammone, R.˜J. (Ed.), Neural Networks for Speech and Image Processing. Chapman-Hall, pp. 126–142.

[40] Politis, D., Romano, J., Wolf, M., 1999. Subsampling. Springer-Verlang, New York.

[41] Quinlan, J.˜R., 1996. Bagging, boosting, and C4.5. In: Proc. 13th National Conference on Artificial Intelligence. Cambridge, MA, pp. 725–730.

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

[43] Reid, S., Grudic, G., 2009. Regularized linear models in stacked generalization. In: Benediktsson, J.˜A., Kittler, J., Roli, F. (Eds.), Multiple Classifier Systems. Vol. 5519 of Lecture Notes in Computer Science. Springer, pp. 112–121.

[44] Rodríguez, J.˜J., Kuncheva, L.˜I., Alonso, C.˜J., 2006. Rotation forest: A new classifier ensemble method. IEEE Transactions on Pattern Analysis and Machine Intelligence 28˜(10), 1619–1630.

[45] Ruta, D., Gabrys, B., Mar. 2005. Classifier selection for majority voting. Information Fusion 6˜(1), 63–81.

[46] Schapire, R.˜E., 1990. The strength of weak learnability. Machine Learning 5, 197–227.

[47] Tibshirani, R., 1996. Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society. Series B (Methodological) 58˜(1), 267–288.

[48] Ting, K.˜M., Witten, I.˜H., 1999. Issues in stacked generalization. Journal of Artificial Intelligence Research 10, 271–289.

[49] Ueda, N., Nakano, R., 1996. Generalization error of ensemble estimators. In: Proceedings of International Conference on Neural Networks. pp. 90–95.

[50] Venables, W.˜N., Ripley, B.˜D., 2002. Modern Applied Statistics with S, 4th Edition. Springer, New York, iSBN 0-387-95457-0.

[51] Wilcoxon, F., 1968. Individual comparisons by ranking methods. Biometrics Bulletin 1˜(6), 80–83.

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

[53] Wolpert, D.˜H., 1992. Stacked generalization. Neural Networks 5˜(2), 241–259.

[54] Zhang, C.-X., Zhang, J.-S., 2008. Rotboost: A technique for combining rotation forest and adaboost. Pattern Recognition Letters 29˜(10), 1524–1536.

[55] Zhang, Y., Burer, S., Street, W.˜N., 2006. Ensemble pruning via semi-definite programming. Journal of Machine Learning Research 7, 1315–1338.

[56] Zhou, Z.-H., Wu, J., Tang, W., 2002. Ensembling neural networks: Many could be better than all. Artificial Intelligence 137˜(1-2), 239–263.

[57] Zou, H., Hastie, T., 2005. Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society B 67, 301–320.