Journal of Multivariate Analysis 146 (2016) 18–28

Contents lists available at ScienceDirect

Journal of Multivariate Analysis

journal homepage: www.elsevier.com/locate/jmva

COBRA: A combined regression strategy

Gérard Biau [a,b], Aurélie Fischer [c], Benjamin Guedj [d,*], James D. Malley [e]

[a] Université Pierre et Marie Curie, France
[b] Institut universitaire de France, France
[c] Université Paris Diderot, France
[d] Inria, France
[e] National Institutes of Health, USA

ARTICLE INFO

Article history:
Received 21 October 2014
Available online 21 April 2015

AMS 2010 subject classifications:
62G05
62G20

Keywords:
Combining estimators
Consistency
Nonlinearity
Nonparametric regression
Prediction

ABSTRACT

A new method for combining several initial estimators of the regression function is introduced. Instead of building a linear or convex optimized combination over a collection of basic estimators $r_1, \ldots, r_M$, we use them as a collective indicator of the proximity between the training data and a test observation. This local distance approach is model-free and very fast. More specifically, the resulting nonparametric/nonlinear combined estimator is shown to perform asymptotically at least as well in the L^2 sense as the best combination of the basic estimators in the collective. A companion R package called COBRA (standing for COmBined Regression Alternative) is presented (downloadable on http://cran.r-project.org/web/packages/COBRA/index.html). Substantial numerical evidence is provided on both synthetic and real data sets to assess the excellent performance and velocity of our method in a large variety of prediction problems.

© 2015 Elsevier Inc. All rights reserved.

1. Introduction

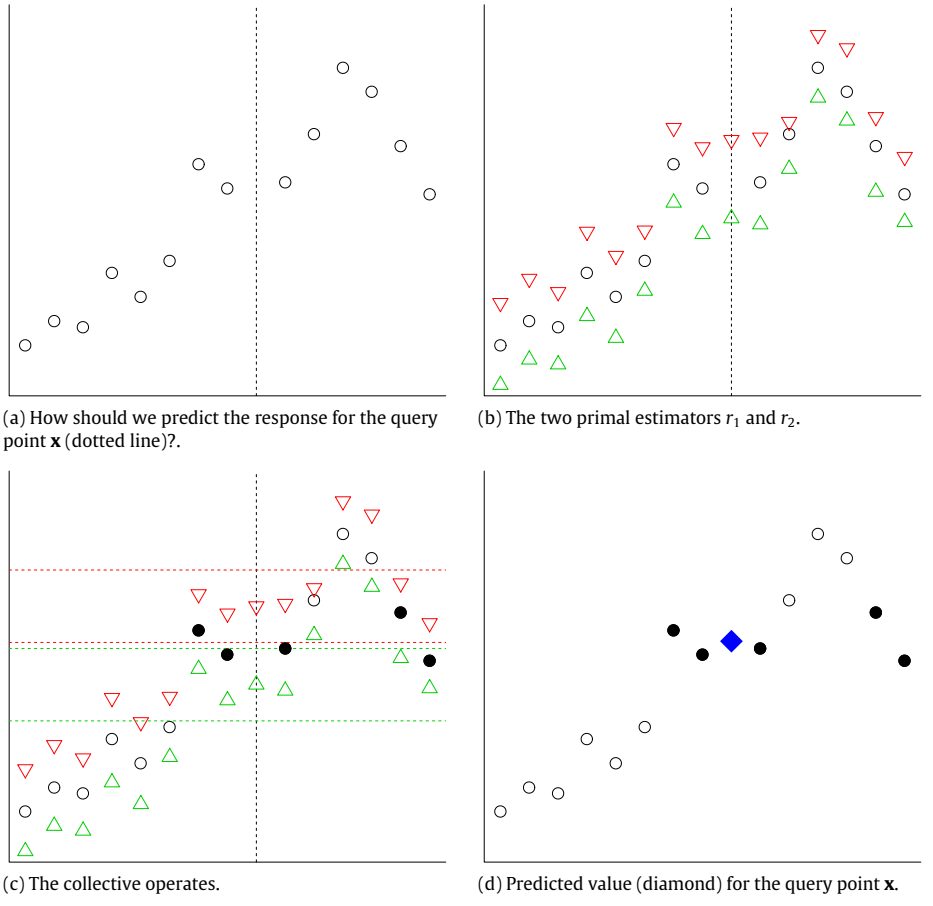
Recent years have witnessed a growing interest in combined statistical procedures, supported by a considerable research and extensive empirical evidence. Indeed, the increasing number of available estimation and prediction methods (hereafter denoted *machines*) in a wide range of modern statistical problems naturally suggests using some efficient strategy for combining procedures and estimators. Such an approach would be a valuable research and development tool, for example when dealing with high or infinite dimensional data.

There exists an extensive literature on linear aggregation of estimators, in a wide range of statistical models: A review of these methods may be found for example in [9]. Our contribution relies on a nonparametric/nonlinear approach based on an original proximity criterion to combine estimators. In that sense, it is different from existing techniques.

Indeed, the present article investigates a novel point of view, motivated by the sense that nonlinear, data-dependent techniques are a source of analytic flexibility. Instead of forming a linear combination of estimators, we propose an original nonlinear method for combining the outcomes over some list of candidate procedures. We call this combined scheme a regression collective over the given basic machines. We consider the problem of building a new estimator by combining M estimators of the regression function, thereby exploiting an idea proposed in the context of supervised classification by Mojirsheibani [19]. Given a set of preliminary estimators $r_1, \ldots, r_M$, the idea behind this combining method is an "unanimity" concept, which is based on the values predicted by $r_1, \ldots, r_M$ for the data and for a new observation $\mathbf{x}$. In a

* Corresponding author.
E-mail address: benjamin.guedj@inria.fr (B. Guedj).

http://dx.doi.org/10.1016/j.jmva.2015.04.007
0047-259X/© 2015 Elsevier Inc. All rights reserved.



(a) How should we predict the response for the query point **x** (dotted line)?.

(b) The two primal estimators r_1 and r_2.

(c) The collective operates.

(d) Predicted value (diamond) for the query point **x**.

Fig. 1. A toy example: Combining two primal estimators.

nutshell, a data point is considered to be "close" to **x**, and consequently, reliable for contributing to the estimation of this new observation, if all estimators predict values which are close to each other for **x** and this data item, *i.e.*, not more distant than a prespecified threshold ε. The predicted value corresponding to this query point **x** is then set to the average of the responses of the selected observations. Let us stress here that the average is over the original outcome values of the selected observations, and *not* over the estimates provided by the several machines for these observations.

To make the concept clear, consider the following toy example illustrated by Fig. 1. Assume we are given the observations plotted in circles, and the values predicted by two known machines r_1 and r_2 (triangles pointing up and down, respectively). The goal is to predict the response for the new point **x** (along the dotted line). Setting a threshold ε, the black solid circles are the data points $(\mathbf{x}_i, y_i)$ within the two dotted intervals, *i.e.*, such that for $m = 1, 2$, $|r_m(\mathbf{x}_i) - r_m(\mathbf{x})| \leq \varepsilon$. Averaging the corresponding y_i's yields the prediction for **x** (diamond).

We stress that the central and original idea behind our approach is that the resulting regression predictor is a nonlinear, nonparametric, data-dependent function of the basic predictors $r_1, \ldots, r_M$, where the predictors are used to determine a local distance between a new test instance and the original training data. To the best of our knowledge there exists no formalized procedure in the machine learning and aggregation literature that operates as ours does. In particular, note that the original nonparametric nature of our combined estimator opens up new perspectives of research.

Indeed, though we have in mind a batch setting where the data collected consists in an n-sample of i.i.d. replications of some variable $(\mathbf{X}, Y)$, our procedure may be linked to other situations. For example, consider the case of functional data analysis (see [8], and [3], for a survey on recent developments). Even though our method is fitted for finite dimensional data, it may be naturally extended to functional data after a suitable preprocessing of the curves. For example, this can be achieved using an expansion of the curves on an appropriate functional dictionary, and/or *via* a variable selection approach, as in [1]. Note that in a recent work, Cholaquidis et al. [4] adapts our procedure in a classification setting, also in a functional example.

Along with this paper, we release the software `COBRA` [11] which implements the method as an additional package to the statistical software R (see [23]). `COBRA` is freely downloadable on the CRAN website.[1] As detailed in Section 3, we undertook

[1] http://cran.r-project.org/web/packages/COBRA/index.html.

a lengthy series of numerical experiments, over which COBRA proved extremely successful. These stunning results lead us to believe that regression collectives can provide valuable insights on a wide range of prediction problems. Further, these same results demonstrate that COBRA has remarkable speed in terms of CPU timings. In the context of high-dimensional (such as genomic) data, such velocity is critical, and in fact COBRA can natively take advantage of multi-core parallel environments.

The paper is organized as follows. In Section 2, we describe the combined estimator – the regression collective – and derive a nonasymptotic risk bound. Next we present the main result, that is, the collective is asymptotically at least as good as any functional of the basic estimators. We also provide a rate of convergence for our procedure. Section 3 is devoted to the companion R package COBRA and presents benchmarks of its excellent performance on both simulated and real data sets, including high-dimensional models. We also show that COBRA compares favorably with two competitors, Super Learner [25] and the exponentially weighted aggregate (see for example [9]), in that it performs similarly in most situations, much better in some, while it is consistently faster than the Super Learner in every case. Finally, for ease of exposition, proofs and additional simulation results (figures and tables with (SM) as suffix) are postponed to a Supplementary material (see Appendix A).

2. The combined estimator

2.1. Notation

Throughout the article, we assume that we are given a training sample denoted by $\mathscr{D}_n = \{(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)\}$. $\mathscr{D}_n$ is composed of i.i.d. random variables taking their values in $\mathbb{R}^d \times \mathbb{R}$, and distributed as an independent prototype pair $(\mathbf{X}, Y)$ satisfying $\mathbb{E}Y^2 < \infty$ (with the notation $\mathbf{X} = (X_1, \ldots, X_d)$). The space $\mathbb{R}^d$ is equipped with the standard Euclidean metric. Our goal is to consistently estimate the regression function $r^\star(\mathbf{x}) = \mathbb{E}[Y|\mathbf{X} = \mathbf{x}]$, $\mathbf{x} \in \mathbb{R}^d$, using the data $\mathscr{D}_n$.

To begin with, the original data set $\mathscr{D}_n$ is split into two data sequences $\mathscr{D}_k = \{(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_k, Y_k)\}$ and $\mathscr{D}_\ell = \{(\mathbf{X}_{k+1}, Y_{k+1}), \ldots, (\mathbf{X}_n, Y_n)\}$, with $\ell = n - k \geq 1$. For ease of notation, the elements of $\mathscr{D}_\ell$ are renamed $\{(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_\ell, Y_\ell)\}$. There is a slight abuse of notation here, as the same letter is used for both subsets $\mathscr{D}_k$ and $\mathscr{D}_\ell$— however, this should not cause any trouble since the context is clear.

Now, suppose that we are given a collection of $M \geq 1$ competing candidates $r_{k,1}, \ldots, r_{k,M}$ to estimate $r^\star$. These basic estimators – basic machines – are assumed to be generated using only the first subsample $\mathscr{D}_k$. These machines can be any among the researcher's favorite toolkit, such as linear regression, kernel smoother, SVM, Lasso, neural networks, naive Bayes, or random forests. They could equally well be any ad hoc regression rules suggested by the experimental context. The essential idea is that these basic machines can be parametric, nonparametric, or semi-parametric, with possible tuning rules. All that is asked for is that each of the $r_{k,m}(\mathbf{x})$, $m = 1, \ldots, M$, is able to provide an estimation of $r^\star(\mathbf{x})$ on the basis of $\mathscr{D}_k$ alone. Thus, any collection of model-based or model-free machines are allowed, and our way of combining such a collection is here called the regression collective. Let us emphasize that the number of basic machines M is considered as fixed throughout this paper. Hence, the number of machines is not expected to grow and is typically of a reasonable size (M is chosen on the order of 10 in Section 3).

Given the collection of basic machines $\mathbf{r}_k = (r_{k,1}, \ldots, r_{k,M})$, we define the collective estimator T_n to be

$$T_n\left(\mathbf{r}_k(\mathbf{x})\right) = \sum_{i=1}^{\ell} W_{n,i}(\mathbf{x}) Y_i, \quad \mathbf{x} \in \mathbb{R}^d,$$

where the random weights $W_{n,i}(\mathbf{x})$ take the form

$$W_{n,i}(\mathbf{x}) = \frac{\mathbf{1}_{\bigcap_{m=1}^{M}\{|r_{k,m}(\mathbf{x}) - r_{k,m}(\mathbf{X}_i)| \leq \varepsilon_\ell\}}}{\sum_{j=1}^{\ell} \mathbf{1}_{\bigcap_{m=1}^{M}\{|r_{k,m}(\mathbf{x}) - r_{k,m}(\mathbf{X}_j)| \leq \varepsilon_\ell\}}}. \tag{2.1}$$

In this definition, ε_ℓ is some positive parameter and, by convention, $0/0 = 0$.

The weighting scheme used in our regression collective is distinctive but not obvious. Starting from [7] and [12], we see that T_n is a local averaging estimator in the following sense: The predicted value for $r^\star(\mathbf{x})$, that is, the estimated outcome at the query point $\mathbf{x}$, is the unweighted average over those Y_i's such that $\mathbf{X}_i$ is "close" to the query point. More precisely, for each $\mathbf{X}_i$ in the sample $\mathscr{D}_\ell$, "close" means that the output at the query point, generated from each basic machine, is within an ε_ℓ-distance of the output generated by the same basic machine at $\mathbf{X}_i$. If a basic machine evaluated at $\mathbf{X}_i$ is close to the basic machine evaluated at the query point $\mathbf{x}$, then the corresponding outcome Y_i is included in the average, and not otherwise. Also, as a further note of clarification: "Closeness" of the $\mathbf{X}_i$'s is not here to be understood in the Euclidean sense. It refers to closeness of the primal estimators outputs at the query point as compared to the outputs over all points in the training data. Training points $\mathbf{X}_i$ that are close, in this sense, to the corresponding outputs at the query point contribute to the indicator function for the corresponding outcome Y_i. This alternative approach is motivated by the fact that a major issue in learning problems consists of devising a metric that is suited to the data (see, *e.g.*, the monograph by Pekalska and Duin [20]).

In this context, ε_ℓ plays the role of a smoothing parameter: Put differently, in order to retain Y_i, all basic estimators $r_{k,1}, \ldots, r_{k,M}$ have to deliver predictions for the query point $\mathbf{x}$ which are in a ε_ℓ-neighborhood of the predictions $r_{k,1}(\mathbf{X}_i), \ldots, r_{k,M}(\mathbf{X}_i)$. Note that the greater ε_ℓ, the more tolerant the process. It turns out that the practical performance

of T_n strongly relies on an appropriate choice of ε_ℓ. This important question will be discussed in Section 3, where we devise an automatic (*i.e.*, data-dependent) selection strategy of ε_ℓ.

Next, we note that the subscript n in T_n may be a little confusing, since T_n is a weighted average of the Y_i's in $\mathscr{D}_\ell$ only. However, T_n depends on the entire data set $\mathscr{D}_n$, as the rest of the data is used to set up the original machines $r_{k,1}, \ldots, r_{k,M}$. Most importantly, it should be noticed that the combined estimator T_n is nonlinear with respect to the basic estimators $r_{k,m}$. As such, it is inspired by the preliminary work of Mojirsheibani [19] in the supervised classification context.

In addition, let us mention that, in the definition of the weights (2.1), all original estimators are invited to have the same, equally valued opinion on the importance of the observation $\mathbf{X}_i$ (within the range of ε_ℓ) for the corresponding Y_i to be integrated in the combination T_n. However, this unanimity constraint may be relaxed by imposing, for example, that a fixed fraction $\alpha \in \{1/M, 2/M, \ldots, 1\}$ of the machines agrees on the importance of $\mathbf{X}_i$. In that case, the weights take the more sophisticated form

$$W_{n,i}(\mathbf{x}) = \frac{\mathbf{1}_{\left\{\sum_{m=1}^M \mathbf{1}_{\{|r_{k,m}(\mathbf{x}) - r_{k,m}(\mathbf{X}_i)| \le \varepsilon_\ell\}} \ge M\alpha\right\}}}{\sum_{j=1}^{\ell} \mathbf{1}_{\left\{\sum_{m=1}^M \mathbf{1}_{\{|r_{k,m}(\mathbf{x}) - r_{k,m}(\mathbf{X}_j)| \le \varepsilon_\ell\}} \ge M\alpha\right\}}}.$$

It turns out that adding the parameter α does not change the asymptotic properties of T_n, provided $\alpha \to 1$. Thus, to keep a sufficient degree of clarity in the mathematical statements and subsequent proofs, we have decided to consider only the case $\alpha = 1$ (*i.e.*, unanimity). Extension of the results to more general values of α is left for future work. On the other hand, as highlighted by Section 3, α has a nonnegligible impact on the performance of the combined estimator. Accordingly, we will discuss in Section 3 an automatic procedure to select this extra parameter.

2.2. Theoretical performance

This section is devoted to the study of some asymptotic and nonasymptotic properties of the combined estimator T_n, whose quality will be assessed by the quadratic risk

$$\mathbb{E}\left|T_n\left(\mathbf{r}_k(\mathbf{X})\right) - r^\star(\mathbf{X})\right|^2.$$

Here and later, $\mathbb{E}$ denotes the expectation with respect to both $\mathbf{X}$ and the sample $\mathscr{D}_n$. Everywhere in the document, it is assumed that $\mathbb{E}|r_{k,m}(\mathbf{X})|^2 < \infty$ for all $m = 1, \ldots, M$.

For any $m = 1, \ldots, M$, let $r_{k,m}^{-1}$ denote the inverse image of machine $r_{k,m}$. Assume that for any $m = 1, \ldots, M$,

$$r_{k,m}^{-1}((t, +\infty)) \underset{t\uparrow+\infty}{\searrow} \emptyset \quad \text{and} \quad r_{k,m}^{-1}((-\infty, t)) \underset{t\downarrow-\infty}{\searrow} \emptyset. \tag{2.2}$$

It is stressed that this is a mild assumption which is met, for example, whenever the machines are bounded. Throughout, we let

$$T\left(\mathbf{r}_k(\mathbf{X})\right) = \mathbb{E}\left[Y|\mathbf{r}_k(\mathbf{X})\right]$$

and note that, by the very definition of the L^2 conditional expectation,

$$\mathbb{E}\left|T(\mathbf{r}_k(\mathbf{X})) - Y\right|^2 \le \inf_f \mathbb{E}\left|f(\mathbf{r}_k(\mathbf{X})) - Y\right|^2, \tag{2.3}$$

where the infimum is taken over all square integrable functions of $\mathbf{r}_k(\mathbf{X})$.

Our first result is a nonasymptotic inequality, which states that the combined estimator behaves as well as the best one in the original list, within a term measuring how far T_n is from T.

Proposition 2.1. *Let $\mathbf{r}_k = (r_{k,1}, \ldots, r_{k,M})$ be the collection of basic estimators, and let $T_n(\mathbf{r}_k(\mathbf{x}))$ be the combined estimator. Then, for all distributions of $(\mathbf{X}, Y)$ with $\mathbb{E}Y^2 < \infty$,*

$$\mathbb{E}|T_n(\mathbf{r}_k(\mathbf{X})) - r^\star(\mathbf{X})|^2 \le \mathbb{E}|T_n(\mathbf{r}_k(\mathbf{X})) - T(\mathbf{r}_k(\mathbf{X}))|^2 + \inf_f \mathbb{E}|f(\mathbf{r}_k(\mathbf{X})) - r^\star(\mathbf{X})|^2,$$

where the infimum is taken over all square integrable functions of $\mathbf{r}_k(\mathbf{X})$. In particular,

$$\mathbb{E}|T_n(\mathbf{r}_k(\mathbf{X})) - r^\star(\mathbf{X})|^2 \le \min_{m=1,\ldots,M} \mathbb{E}|r_{k,m}(\mathbf{X}) - r^\star(\mathbf{X})|^2 + \mathbb{E}|T_n(\mathbf{r}_k(\mathbf{X})) - T(\mathbf{r}_k(\mathbf{X}))|^2.$$

Proposition 2.1 guarantees the performance of T_n with respect to the basic machines, whatever the distribution of $(\mathbf{X}, Y)$ is and regardless of which initial estimator is actually the best. The term $\min_{m=1,\ldots,M} \mathbb{E}|r_{k,m}(\mathbf{X}) - r^\star(\mathbf{X})|^2$ may be regarded as a bias term, whereas the term $\mathbb{E}|T_n(\mathbf{r}_k(\mathbf{X})) - T(\mathbf{r}_k(\mathbf{X}))|^2$ is a variance-type term, which can be asymptotically neglected, as shown by the following result.

Proposition 2.2. *Assume that* $\varepsilon_\ell \to 0$ *and* $\ell\varepsilon_\ell^M \to \infty$ *as* $\ell \to \infty$. *Then*

$$\mathbb{E}\left|T_n\left(\mathbf{r}_k(\mathbf{X})\right) - T\left(\mathbf{r}_k(\mathbf{X})\right)\right|^2 \to 0 \quad \text{as } \ell \to \infty,$$

for all distributions of $(\mathbf{X}, Y)$ *with* $\mathbb{E}Y^2 < \infty$. *Thus,*

$$\limsup_{\ell\to\infty} \mathbb{E}\left|T_n\left(\mathbf{r}_k(\mathbf{X})\right) - r^\star(\mathbf{X})\right|^2 \le \inf_f \mathbb{E}\left|f\left(r_{k,m}(\mathbf{X})\right) - r^\star(\mathbf{X})\right|^2.$$

In particular,

$$\limsup_{\ell\to\infty} \mathbb{E}\left|T_n\left(\mathbf{r}_k(\mathbf{X})\right) - r^\star(\mathbf{X})\right|^2 \le \min_{m=1,\dots,M} \mathbb{E}\left|r_{k,m}(\mathbf{X}) - r^\star(\mathbf{X})\right|^2.$$

This result is remarkable, for two reasons. Firstly, it shows that, in terms of predictive quadratic risk, the combined estimator does asymptotically at least as well as the best primitive machine. Secondly, the result is nearly universal, in the sense that it is true for all distributions of $(\mathbf{X}, Y)$ such that $\mathbb{E}Y^2 < \infty$.

This is especially interesting because the performance of any estimation procedure eventually depends upon some model and smoothness assumptions on the observations. For example, a linear regression fit performs well if the distribution is truly linear, but may behave poorly otherwise. Similarly, the Lasso procedure is known to do a good job for non-correlated designs, with no clear guarantee however in adversarial situations. Likewise, performance of nonparametric procedures such as the k-nearest neighbor method, kernel estimators and random forests dramatically deteriorate as the ambient dimension increases, but may be significantly improved if the true underlying dimension is reasonable. Note that this phenomenon is thoroughly analyzed for the random forests algorithm in [2].

The result exhibited in Proposition 2.2 holds under a minimal regularity assumption on the basic machines. However, this universality comes at a price since we have no guarantee on the rate of convergence of the variance term. Nevertheless, assuming some light additional smoothness conditions, one has the following result, which is the central statement of the paper.

Theorem 2.1. *Assume that* Y *and the basic machines* $\mathbf{r}_k$ *are bounded by some constant* R. *Assume moreover that there exists a constant* $L \ge 0$ *such that, for every* $k \ge 1$,

$$|T(\mathbf{r}_k(\mathbf{x})) - T(\mathbf{r}_k(\mathbf{y}))| \le L|\mathbf{r}_k(\mathbf{x}) - \mathbf{r}_k(\mathbf{y})|, \quad \mathbf{x}, \mathbf{y} \in \mathbb{R}^d.$$

Then, with the choice $\varepsilon_\ell \propto \ell^{-\frac{1}{M+2}}$, *one has*

$$\mathbb{E}\left|T_n\left(\mathbf{r}_k(\mathbf{X})\right) - r^\star(\mathbf{X})\right|^2 \le \min_{m=1,\dots,M} \mathbb{E}\left|r_{k,m}(\mathbf{X}) - r^\star(\mathbf{X})\right|^2 + C\ell^{-\frac{2}{M+2}},$$

for some positive constant $C = C(R, L)$, *independent of* k.

Theorem 2.1 offers an oracle-type inequality with leading constant 1 (*i.e.*, sharp oracle inequality), stating that the risk of the regression collective is bounded by the lowest risk among those of the basic machines, *i.e.*, our procedure mimics the performance of the oracle over the set $\{r_{k,m}: m = 1, \dots, M\}$, plus a remainder term of the order of $\ell^{-2/(M+2)}$ which is the price to pay for combining M estimators. In our setting, it is important to observe that this term has a limited impact. As a matter of fact, since the number of basic machines M is assumed to be fixed and not too large (the implementation presented in Section 3 considers M at most 6), the remainder term is negligible compared to the standard nonparametric rate $\ell^{-2/(d+2)}$ in dimension d. While the rate $\ell^{-2/(d+2)}$ is affected by the curse of dimensionality when d is large, this is not the case for the term $\ell^{-2/(M+2)}$. That way, our procedure appears well armed to face high dimensional problems. When $d \gg n$, many methods deteriorate and suffer from the curse of dimensionality. However, it is important to note here that even if some of the basic machines $r_{k,1}, \dots, r_{k,M}$ might be less performant in that context, this does not affect in any way our combining procedure. Indeed, forming the regression collective T_n does not require any additional effort if d grows. Obviously, when d is large, the best choice would be to include as basic machines methods and models which are adapted to the high dimensional setting. This is an interesting track for future research, which is connected to functional data analysis and dimension-reduction models (see [10]).

Obviously, under the assumption that the distribution of $(\mathbf{X}, Y)$ might be described parametrically and that one of the initial estimators is adapted to this distribution, faster rates of the order of $1/\ell$ could emerge in the bias term. Nonetheless, the regression collective is designed for much more adversarial regression problems, hence the rate exhibited in Theorem 2.1 appears satisfactory. We stress that our approach carries no assumption on the random design and mild ones over the primal estimators, in line with our attempt to design a procedure which is as model-free as possible.

The central motivation for our method is that model and smoothness assumptions are usually unverifiable, especially in modern high-dimensional and large scale data sets. To circumvent this difficulty, researchers often try many different methods and retain the one exhibiting the best empirical (*e.g.*, cross-validated) results. Our combining strategy offers a nice alternative, in the sense that if one of the initial estimators is consistent for a given class $\mathscr{M}$ of distributions, then, under

light smoothness assumptions, T_n inherits the same property. To be more precise, assume that the initial pool of estimators includes a consistent estimator, *i.e.*, that one of the original estimators, say r_{k,m_0}, satisfies

$$\mathbb{E}\left|r_{k,m_0}(\mathbf{X}) - r^\star(\mathbf{X})\right|^2 \to 0 \quad \text{as } k \to \infty,$$

for all distributions of $(\mathbf{X}, Y)$ in some class $\mathscr{M}$. Then, under the assumptions of Theorem 2.1, with the choice $\varepsilon_\ell \propto \ell^{-\frac{1}{M+2}}$, one has

$$\lim_{k,\ell\to\infty} \mathbb{E}\left|T_n\left(\mathbf{r}_k(\mathbf{X})\right) - r^\star(\mathbf{X})\right|^2 = 0.$$

3. Implementation and numerical studies

This section is devoted to the implementation of the described method. Its excellent performance is then assessed in a series of experiments. The companion R package `COBRA` (standing for COmBined Regression Alternative) is available on the CRAN website,[2] for Linux, Mac and Windows platforms (see [11]). Note that in a will to favor its execution speed, `COBRA` includes a `parallel` option, allowing for improved performance on multi-core computers (from [14]).

As raised in the previous section, a precise calibration of the smoothing parameter ε_ℓ is crucial. Clearly, a value that is too small will discard many machines and most weights will be zero. Conversely, a large value sets all weights to $1/\Sigma$ with

$$\Sigma = \sum_{j=1}^{\ell} \mathbf{1}_{\bigcap_{m=1}^{M}\{|r_{k,m}(\mathbf{x}) - r_{k,m}(\mathbf{X}_j)| \le \varepsilon_\ell\}},$$

giving the naive predictor that does not account for any new data point and predicts the mean over the sample $\mathscr{D}_\ell$. We also consider a relaxed version of the unanimity constraint: Instead of requiring global agreement over the implemented machines, consider some $\alpha \in (0, 1]$ and keep observation Y_i in the construction of T_n if and only if at least a proportion α of the machines agrees on the importance of $\mathbf{X}_i$. This parameter requires some calibration. To understand this better, consider the following toy example: On some data set, assume most machines but one have nice predictive performance. For any new data point, requiring global agreement will fail since the pool of machines is heterogeneous. In this regard, α should be seen as a measure of homogeneity: If a small value is selected, it may be an indicator that some machines perform (possibly much) better than some others. Conversely, a large value indicates that the predictive abilities of the machines are close.

A natural measure of the risk in the prediction context is the empirical quadratic loss, namely

$$\hat{R}(\hat{\mathbf{Y}}) = \frac{1}{p}\sum_{j=1}^{p}(\hat{Y}_j - Y_j)^2,$$

where $\hat{\mathbf{Y}} = (\hat{Y}_1, \ldots, \hat{Y}_p)$ is the vector of predicted values for the responses $Y_1, \ldots, Y_p$ and $\{(\mathbf{X}_j, Y_j)\}_{j=1}^p$ is a testing sample. We adopted the following protocol: Using a simple data-splitting device, ε_ℓ and α are chosen by minimizing the empirical risk $\hat{R}$ over the set $\{\varepsilon_{\ell,\min}, \ldots, \varepsilon_{\ell,\max}\} \times \{1/M, \ldots, 1\}$, where $\varepsilon_{\ell,\min} = 10^{-300}$ and $\varepsilon_{\ell,\max}$ is proportional to the largest absolute difference between two predictions of the pool of machines.

In the package, the number $\#\{\varepsilon_{\ell,\min}, \ldots, \varepsilon_{\ell,\max}\}$ of evaluated values may be modified by the user, otherwise the default value 200 is chosen. It is also possible to choose either a linear or a logistic scale. Fig. 2 (SM) illustrates the discussion about the choice of ε_ℓ and α.

By default, `COBRA` includes the following classical packages dealing with regression estimation and prediction. However, note that the user has the choice to modify this list to her/his own convenience:

- Lasso (R package `lars`, see [13]).
- Ridge regression (R package `ridge`, see [6]).
- k-nearest neighbors (R package `FNN`, see [15]).
- CART algorithm (R package `tree`, see [24]).
- Random Forests algorithm (R package `randomForest`, see [16]).

First, `COBRA` is benchmarked on synthetic data. For each of the following eight models, two designs are considered: Uniform over $(-1, 1)^d$ (referred to as "Uncorrelated" in Tables 1–3), and Gaussian with mean 0 and covariance matrix Σ with $\Sigma_{ij} = 2^{-|i-j|}$ ("Correlated"). Models considered cover a wide spectrum of contemporary regression problems. Indeed, Model 1 is a toy example, Model 2 comes from [25], Models 3 and 4 appear in [18]. Model 5 is somewhat a classic setting. Model 6 is about predicting labels, Model 7 is inspired by high-dimensional sparse regression problems. Finally, Model 8 deals with probability estimation, forming a link with nonparametric model-free approaches such as in [17]. In the sequel, we let $\mathscr{N}(\mu, \sigma^2)$ denote a Gaussian random variable with mean μ and variance σ^2. In the simulations, the training data set was usually set to 80% of the whole sample, then split into two equal parts corresponding to $\mathscr{D}_k$ and $\mathscr{D}_\ell$.

Model 1. $n = 800$, $d = 50$, $Y = X_1^2 + \exp(-X_2^2)$.

Model 2. $n = 600$, $d = 100$, $Y = X_1X_2 + X_3^2 - X_4X_7 + X_8X_{10} - X_6^2 + \mathscr{N}(0, 0.5)$.

[2] http://cran.r-project.org/web/packages/COBRA/index.html.

Table 1
Quadratic errors of the implemented machines and COBRA. Means and standard deviations over 100 independent replications.

		lars	ridge	fnn	tree	rf	COBRA
Uncorr.							
Model 1	m.	0.1561	0.1324	0.1585	0.0281	0.0330	**0.0259**
	sd.	0.0123	0.0094	0.0123	0.0043	0.0033	0.0036
Model 2	m.	0.4880	0.2462	0.3070	0.1746	**0.1366**	0.1645
	sd.	0.0676	0.0233	0.0303	0.0270	0.0161	0.0207
Model 3	m.	0.2536	0.5347	1.1603	0.4954	0.4027	**0.2332**
	sd.	0.0271	0.4469	0.1227	0.0772	0.0558	0.0272
Model 4	m.	7.6056	6.3271	10.5890	3.7358	3.5262	**3.3640**
	sd.	0.9419	1.0800	0.9404	0.8067	0.3223	0.5178
Model 5	m.	0.2943	0.3311	0.5169	0.2918	0.2234	**0.2060**
	sd.	0.0214	0.1012	0.0439	0.0279	0.0216	0.0210
Model 6	m.	0.8438	1.0303	2.0702	2.3476	1.3354	**0.8345**
	sd.	0.0916	0.4840	0.2240	0.2814	0.1590	0.1004
Model 7	m.	1.0920	0.5452	0.9459	0.3638	0.3110	**0.3052**
	sd.	0.2265	0.0920	0.0833	0.0456	0.0325	0.0298
Model 8	m.	0.1308	0.1279	0.2243	0.1715	0.1236	**0.1021**
	sd.	0.0120	0.0161	0.0189	0.0270	0.0100	0.0155
Corr.							
Model 1	m.	2.3736	1.9785	2.0958	0.3312	0.5766	**0.3301**
	sd.	0.4108	0.3538	0.3414	0.1285	0.1914	0.1239
Model 2	m.	8.1710	4.0071	4.3892	**1.3609**	1.4768	1.3612
	sd.	1.5532	0.6840	0.7190	0.4647	0.4415	0.4654
Model 3	m.	6.1448	6.0185	8.2154	4.3175	4.0177	**3.7917**
	sd.	11.9450	12.0861	13.3121	11.7386	12.4160	11.1806
Model 4	m.	60.5795	42.2117	51.7293	**9.6810**	14.7731	9.6906
	sd.	11.1303	9.8207	10.9351	3.9807	5.9508	3.9872
Model 5	m.	6.2325	7.1762	10.1254	3.1525	4.2289	**2.1743**
	sd.	2.4320	3.5448	3.1190	2.1468	2.4826	1.6640
Model 6	m.	1.2765	1.5307	2.5230	2.6185	1.2027	**0.9925**
	sd.	0.1381	0.9593	0.2762	0.3445	0.1600	0.1210
Model 7	m.	20.8575	4.4367	5.8893	3.6865	**2.7318**	2.9127
	sd.	7.1821	1.0770	1.2226	1.0139	0.8945	0.9072
Model 8	m.	0.1366	0.1308	0.2267	0.1701	0.1226	**0.0984**
	sd.	0.0127	0.0143	0.0179	0.0302	0.0102	0.0144

Model 3. $n = 600$, $d = 100$, $Y = -\sin(2X_1) + X_2^2 + X_3 - \exp(-X_4) + \mathcal{N}(0, 0.5)$.

Model 4. $n = 600$, $d = 100$, $Y = X_1 + (2X_2 - 1)^2 + \sin(2\pi X_3)/(2 - \sin(2\pi X_3)) + \sin(2\pi X_4) + 2\cos(2\pi X_4) + 3\sin^2(2\pi X_4) + 4\cos^2(2\pi X_4) + \mathcal{N}(0, 0.5)$.

Model 5. $n = 700$, $d = 20$, $Y = \mathbf{1}_{\{X_1 > 0\}} + X_2^3 + \mathbf{1}_{\{X_4 + X_6 - X_8 - X_9 > 1 + X_{14}\}} + \exp(-X_2^2) + \mathcal{N}(0, 0.5)$.

Model 6. $n = 500$, $d = 30$, $Y = \sum_{k=1}^{10} \mathbf{1}_{\{X_k^3 < 0\}} - \mathbf{1}_{\{\mathcal{N}(0,1) > 1.25\}}$.

Model 7. $n = 600$, $d = 300$, $Y = X_1^2 + X_2^2 X_3 \exp(-|X_4|) + X_6 - X_8 + \mathcal{N}(0, 0.5)$.

Model 8. $n = 600$, $d = 50$, $Y = \mathbf{1}_{\{X_1 + X_4^3 + X_9 + \sin(X_{12} X_{18}) + \mathcal{N}(0,0.1) > 0.38\}}$.

Table 1 presents the empirical mean quadratic error and standard deviation over 100 independent replications, for each model and design. Bold numbers identify the lowest error, *i.e.*, the apparent best competitor. Boxplots of errors are presented in Figs. 3 (SM) and 4 (SM). Further, Figs. 5 (SM) and 6 (SM) show the predictive capacities of COBRA, and Fig. 7 (SM) depicts its ability to reconstruct the functional dependence over the covariates in the context of additive regression, assessing the striking performance of our approach in a wide spectrum of statistical settings. A persistent and notable fact is that COBRA performs at least as well as the best machine, especially so in Models 3, 5 and 6.

Next, since more and more problems in contemporary statistics involve high-dimensional data, we have tested the abilities of COBRA in that context. As highlighted by Table 4 (SM) and Fig. 8 (SM), the main message is that COBRA is perfectly able to deal with high-dimensional data, provided that it is generated over machines, at least some of which are known to perform well in such situations (possibly at the price of a sparsity assumption). In that context, we conducted 200 independent replications for the three following models:

Model 9. $n = 500$, $d = 1000$, $Y = X_1 + 3X_3^2 - 2\exp(-X_5) + X_6$. *Uncorrelated design.*

Model 10. $n = 500$, $d = 1000$, $Y = X_1 + 3X_3^2 - 2\exp(-X_5) + X_6$. *Correlated design.*

Table 2
Quadratic errors of `SuperLearner` and `COBRA`. Means and standard deviations over 100 independent replications.

		SL	COBRA
Uncorr.			
Model 1	m.	0.0541	**0.0320**
	sd.	0.0053	0.0104
Model 2	m.	**0.1765**	0.3569
	sd.	0.0167	0.8797
Model 3	m.	**0.2081**	0.2573
	sd.	0.0282	0.0699
Model 4	m.	4.3114	**3.7464**
	sd.	0.4138	0.8746
Model 5	m.	**0.2119**	0.2187
	sd.	0.0317	0.0427
Model 6	m.	**0.7627**	1.0220
	sd.	0.1023	0.3347
Model 7	m.	**0.1705**	0.3103
	sd.	0.0260	0.0490
Model 8	m.	0.1081	**0.1075**
	sd.	0.0121	0.0235
Corr.			
Model 1	m.	0.8733	**0.3262**
	sd.	0.2740	0.1242
Model 2	m.	2.3391	**1.3984**
	sd.	0.4958	0.3804
Model 3	m.	**3.1885**	3.3201
	sd.	1.5101	1.8056
Model 4	m.	25.1073	**9.3964**
	sd.	7.3179	2.8953
Model 5	m.	5.6478	**4.9990**
	sd.	7.7271	9.3103
Model 6	m.	**0.8967**	1.1988
	sd.	0.1197	0.4573
Model 7	m.	**3.0367**	3.1401
	sd.	1.6225	1.6097
Model 8	m.	0.1116	**0.1045**
	sd.	0.0111	0.0216

Model 11. $n = 500, d = 1500, Y = \exp(-X_1) + \exp(X_1) + \sum_{j=2}^{d} X_j^{j/100}$. *Uncorrelated design.*

A legitimate question that arises is where one should cut the initial sample $\mathscr{D}_n$? In other words, for a given data set of size n, what is the optimal value for k? A naive approach is to cut the initial sample in two halves (*i.e.*, $k = n/2$): This appears to be satisfactory provided that n is large enough, which may be too much of an unrealistic assumption in numerous experimental settings. A more involved choice is to adopt a random cut scheme, where k is chosen uniformly in $\{1, \ldots, n\}$. Fig. 9 (SM) presents the boxplots of errors of the five default machines and `COBRA` with that random cutting strategy, and also shows the risk of `COBRA` with respect to k. To illustrate this phenomenon, we tested a thousand random cuts on Model 12. As shown in Fig. 9 (SM), for that particular model, the best value seems to be near $3n/4$.

Model 12. $n = 1200, d = 10, Y = X_1 + 3X_3^2 - 2\exp(-X_5) + X_6$. *Uncorrelated design.*

The average risk of `COBRA` on a thousand replications of Model 12 is 0.3124. Since this delivered a thousand prediction vectors, a natural idea is to take their mean or median. The risk of the mean is 0.2306, and the median has an even better risk (0.2184). Since a random cut scheme may generate some instability, we advise practitioners to compute a few `COBRA` estimators, then compute the mean or median vector of their predictions.

Next, we compare `COBRA` to the Super Learner algorithm [22]. This widely used algorithm was first described in [25] and extended in [21]. Super Learner is used in this section as the key competitor to our method. In a nutshell, the Super Learner trains basic machines $r_1, \ldots, r_M$ on the whole sample $\mathscr{D}_n$. Then, following a V-fold cross-validation procedure, Super Learner adopts a V-blocks partition of the set $\{1, \ldots, n\}$ and computes the matrix

$$H = (H_{ij})_{1 \le i \le n}^{1 \le j \le M},$$

where H_{ij} is the prediction for the query point $\mathbf{X}_i$ made by machine j trained on all remaining $V - 1$ blocks, *i.e.*, excluding the block containing $\mathbf{X}_i$. The Super Learner estimator is then

$$SL = \sum_{j=1}^{M} \hat{\alpha}_j r_j,$$

Table 3
Average CPU-times in seconds. No parallelization. Means and standard deviations over 10 independent replications.

		SL	COBRA
Uncorr.			
Model 1	m.	53.92	**10.92**
	sd.	1.42	0.29
Model 2	m.	57.96	**11.90**
	sd.	0.95	0.31
Model 3	m.	53.70	**10.66**
	sd.	0.55	0.11
Model 4	m.	55.00	**11.15**
	sd.	0.74	0.18
Model 5	m.	28.46	**5.01**
	sd.	0.73	0.06
Model 6	m.	22.97	**3.99**
	sd.	0.27	0.05
Model 7	m.	127.80	**35.67**
	sd.	5.69	1.91
Model 8	m.	32.98	**6.46**
	sd.	1.33	0.33
Corr.			
Model 1	m.	61.92	**11.96**
	sd.	1.85	0.27
Model 2	m.	70.90	**14.16**
	sd.	2.47	0.57
Model 3	m.	59.91	**11.92**
	sd.	2.06	0.41
Model 4	m.	63.58	**13.11**
	sd.	1.21	0.34
Model 5	m.	31.24	**5.02**
	sd.	0.86	0.07
Model 6	m.	24.29	**4.12**
	sd.	0.82	0.15
Model 7	m.	145.18	**41.28**
	sd.	8.97	2.84
Model 8	m.	31.31	**6.24**
	sd.	0.73	0.11

where

$$\hat{\alpha} \in \underset{\alpha \in \Lambda^M}{\arg\inf} \sum_{i=1}^{n} |Y_i - (H\alpha)_i|^2,$$

with Λ^M denoting the simplex

$$\Lambda^M = \left\{ \alpha \in \mathbb{R}^M : \sum_{j=1}^{M} \alpha_j = 1,\ \alpha_j \geq 0 \text{ for any } j = 1, \ldots, M \right\}.$$

This convex aggregation scheme is significantly different from our collective approach. Yet, we feel close to the philosophy carried by the `SuperLearner` package, in that both methods allow the user to aggregate as many machines as desired, then combining them to deliver predictive outcomes. For that reason, it is reasonable to deploy Super Learner as a benchmark in our study of our collective approach.

Table 2 summarizes the performance of `COBRA` and `SuperLearner` (used with `SL.randomForest`, `SL.ridge` and `SL.glmnet`, for the fairness of the comparison) through the described protocol. Both methods compete on similar terms in most models, although `COBRA` proves much more efficient on correlated design in Models 2 and 4. This already remarkable result is to be stressed by the flexibility and velocity showed by `COBRA`. Indeed, as emphasized in Table 3, without even using the `parallel` option, `COBRA` obtains similar or better results than `SuperLearner` roughly five times faster. Note also that `COBRA` suffers from a disadvantage: `SuperLearner` is built on the whole sample $\mathscr{D}_n$ whereas `COBRA` only uses $\ell < n$ data points. Finally, observe that the algorithmic cost of computing the random weights on n_{test} query points is $\ell \times M \times n_{\text{test}}$ operations. In the package, those calculations are handled in C language for optimal speed performance.

Super Learner is a natural competitor on the implementation side. However, on the theoretical side, we do not assume that it should be the only benchmark. Thus, we compared `COBRA` to the popular exponentially weighted aggregate estimator (EWA, see [9]). We implemented the following version of the EWA: For all preliminary estimators $r_{k,1}, \ldots, r_{k,M}$,

their empirical risks $\hat{R}_1, \ldots, \hat{R}_M$ are computed on a subsample of $\mathscr{D}_\ell$ and the EWA is

$$\mathrm{EWA}_\beta : \mathbf{x} \mapsto \sum_{j=1}^{M} \hat{w}_j r_{k,j}(\mathbf{x}), \quad \mathbf{x} \in \mathbb{R}^d,$$

where

$$\hat{w}_j = \frac{\exp(-\beta \hat{R}_j)}{\sum_{i=1}^{M} \exp(-\beta \hat{R}_i)}, \quad j = 1, \ldots, M.$$

The temperature parameter $\beta > 0$ is selected by minimizing the empirical risk of EWA_β over a data-based grid, in the same spirit as the selection of ε_ℓ and α. We conducted 200 independent replications, on Models 9–12. The conclusion is that COBRA outperforms the EWA estimator in some models, and delivers similar performance in others, as shown in Fig. 10 (SM) and Table 5 (SM).

Finally, COBRA is used to process the following real-life data sets:

- Concrete Slump Test[3] (see [27]).
- Concrete Compressive Strength[4] (see [26]).
- Wine Quality[5] (see [5]). We point out that the Wine Quality data set involves supervised classification and leads naturally to a line of future research using COBRA over probability machines (see [17]).

The good predictive performance of COBRA is summarized in Fig. 11 (SM) and errors are presented in Fig. 12 (SM). For every data set, the sample is divided into a training set (90%) and a testing set (10%) on which the predictive performance is evaluated. Boxplots are obtained by randomly shuffling the data points a hundred times.

As a conclusion to this through experimental protocol, it is our belief that COBRA sets a new high standard of reference, a benchmark procedure, both in terms of performance and velocity, for prediction-oriented problems in the context of regression, including high-dimensional problems.

Acknowledgments

The authors thank the editor and two anonymous referees for providing constructive and helpful remarks, thus greatly improving the paper.

Appendix A. Supplementary material

Supplementary material related to this article can be found online at http://dx.doi.org/10.1016/j.jmva.2015.04.007.

References

[1] G. Aneiros, P. Vieu, Variable selection in infinite-dimensional problems, Statist. Probab. Lett. 94 (2014) 12–20.
[2] G. Biau, Analysis of a random forests model, J. Mach. Learn. Res. 13 (2012) 1063–1095.
[3] E.G. Bongiorno, E. Salinelli, A. Goia, P. Vieu, Contributions in Infinite-dimensional Statistics and Related Topics, Società Editrice Esculapio, 2014.
[4] A. Cholaquidis, R. Fraiman, J. Kalemkerian, P. Llop, An nonlinear aggregation type classifier, 2015. Preprint.
[5] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, J. Reis, Modeling wine preferences by data mining from physicochemical properties, Decis. Support. Syst. 47 (2009) 547–553.
[6] E. Cule, ridge: Ridge Regression with automatic selection of the penalty parameter. R package version 2.1-2, 2012. URL http://CRAN.R-project.org/package=ridge.
[7] L. Devroye, L. Györfi, G. Lugosi, A Probabilistic Theory of Pattern Recognition, Springer, 1996.
[8] F. Ferraty, P. Vieu, Nonparametric Functional Data Analysis: Theory and Practice, Springer, 2006.
[9] C. Giraud, Introduction to High-Dimensional Statistics, Chapman & Hall/CRC, 2014.
[10] A. Goia, P. Vieu, A partitioned single functional index model, Comput. Statist. (2014).
[11] B. Guedj, COBRA: COmBined Regression Alternative. R package version 0.99.4, 2013. URL http://cran.r-project.org/web/packages/COBRA/index.html.
[12] L. Györfi, M. Kohler, A. Krzyżak, H. Walk, A Distribution-Free Theory of Nonparametric Regression, Springer, 2002.
[13] T. Hastie, B. Efron, lars: Least Angle Regression, Lasso and Forward Stagewise. R package version 1.1, 2012. URL http://CRAN.R-project.org/package=lars.
[14] J. Knaus, snowfall: Easier cluster computing (based on snow). R package version 1.84, 2010. URL http://CRAN.R-project.org/package=snowfall.
[15] S. Li, FNN: Fast Nearest Neighbor search algorithms and applications. R package version 1.1, 2013. URL http://CRAN.R-project.org/package=FNN.
[16] A. Liaw, M. Wiener, Classification and regression by randomforest, R News 2 (2002) 18–22. URL http://CRAN.R-project.org/doc/Rnews/.
[17] J.D. Malley, J. Kruppa, A. Dasgupta, K.G. Malley, A. Ziegler, Probability machines: Consistent probability estimation using nonparametric learning machines, Methods Inf. Med. 51 (2012) 74–81.
[18] L. Meier, S.A. van de Geer, P. Bühlmann, High-dimensional additive modeling, Ann. Statist. 37 (2009) 3779–3821.
[19] M. Mojirsheibani, Combining classifiers via discretization, J. Amer. Statist. Assoc. 94 (1999) 600–609.

[3] http://archive.ics.uci.edu/ml/datasets/Concrete+Slump+Test.
[4] http://archive.ics.uci.edu/ml/datasets/Concrete+Compressive+Strength.
[5] http://archive.ics.uci.edu/ml/datasets/Wine+Quality.

[20] E. Pekalska, R.P.W. Duin, The Dissimilarity Representation for Pattern Recognition: Foundations and Applications, in: Machine Perception and Artificial Intelligence, vol. 64, World Scientific, 2005.
[21] E.C. Polley, M.J. van der Laan, Super Learner in Prediction. Tech. Rep., UC Berkeley, 2010.
[22] E.C. Polley, M.J. van der Laan, SuperLearner: Super Learner Prediction. R package version 2.0-9, 2012. URL http://CRAN.R-project.org/package=SuperLearner.
[23] R Core Team, 2014. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.
[24] B. Ripley, tree: Classification and regression trees. R package version 1.0-32, 2012. URL http://CRAN.R-project.org/package=tree.
[25] M.J. van der Laan, E.C. Polley, A.E. Hubbard, Super learner, Stat. Appl. Genet. Mol. Biol. 6 (2007).
[26] I.-C. Yeh, Modeling of strength of high performance concrete using artificial neural networks, Cement Concr. Res. 28 (1998) 1797–1808.
[27] I.-C. Yeh, Modeling slump flow of concrete using second-order regressions and artificial neural networks, Cement Concr. Compos. 29 (2007) 474–480.