Mach Learn (2014) 97:155–176
DOI 10.1007/s10994-014-5453-0

Regression conformal prediction with random forests

Ulf Johansson · Henrik Boström · Tuve Löfström · Henrik Linusson

Received: 14 February 2014 / Accepted: 29 May 2014 / Published online: 9 July 2014

Abstract Regression conformal prediction produces prediction intervals that are valid, i.e., the probability of excluding the correct target value is bounded by a predefined confidence level. The most important criterion when comparing conformal regressors is efficiency; the prediction intervals should be as tight (informative) as possible. In this study, the use of random forests as the underlying model for regression conformal prediction is investigated and compared to existing state-of-the-art techniques, which are based on neural networks and k-nearest neighbors. In addition to their robust predictive performance, random forests allow for determining the size of the prediction intervals by using out-of-bag estimates instead of requiring a separate calibration set. An extensive empirical investigation, using 33 publicly available data sets, was undertaken to compare the use of random forests to existing state-of-the-art conformal predictors. The results show that the suggested approach, on almost all confidence levels and using both standard and normalized nonconformity functions, produced significantly more efficient conformal predictors than the existing alternatives.

Keywords Conformal prediction · Random forests · Regression

Editors: Toon Calders, Rosa Meo, Floriana Esposito, and Eyke Hullermeier.

U. Johansson (✉) · T. Löfström · H. Linusson
School of Business and IT, University of Borås, Borås, Sweden
e-mail: ulf.johansson@hb.se

T. Löfström
e-mail: tuve.lofstrom@hb.se

H. Linusson
e-mail: henrik.linusson@hb.se

H. Boström
Department of Computer and Systems Sciences, Stockholm University, Stockholm, Sweden
e-mail: henrik.bostrom@dsv.su.se

1 Introduction

One of the main motivations for using the conformal prediction (CP) framework (Vovk et al. 2006) is that it provides guarantees for the prediction error; the probability of making incorrect predictions is bounded by a user-provided confidence threshold. In contrast to other learning frameworks that provide similar types of guarantees, e.g., PAC learning (Valiant 1984), CP makes it possible to assess the uncertainty of each single prediction. Hence, rather than just providing a bound on the prediction error for the entire distribution, CP allows for providing different bounds for different instances, something which may be very valuable in many practical applications. For example, knowing that the error of a certain model for predicting the stock price is bounded by 100 dollars with 95 % probability, is not as informative as knowing that for the specific stock we are interested in, the prediction error is, in fact, bounded by ten dollars, i.e., this particular stock is actually easier to predict than the average one. Similarly, in the medical domain, it is of course important to be able to assess the confidence in predictions related to individual patients instead of groups of patients.

CP employs some underlying predictive model, which may have been generated by any standard learning algorithm, for obtaining prediction regions rather than single point predictions. A prediction region corresponds to a set of class labels in a classification context, and to an interval in a regression context. A prediction error, in this framework, occurs when the correct label of a (test) instance is not included in the prediction region. The guarantee given by the conformal prediction framework, under the standard i.i.d. assumption, is that the probability of making an error is bounded by a predetermined confidence level. This means that the number of errors can be controlled, typically reducing the error level by increasing the sizes of the prediction regions or vice versa. There is an obvious resemblance of the conformal prediction framework to standard statistical hypothesis testing, where the type I and II errors are controlled by the choice of significance level.

Since all conformal predictors are *valid*, i.e., the probability of excluding the correct label is bounded by the confidence level, the main criterion when comparing different conformal predictors is their *efficiency*, i.e., the sizes of output prediction regions. Efficiency is, for classification, often measured as the (average) number of labels present in the prediction sets, and for regression as the (average) size of the intervals.

CP relies on real-valued functions, called *nonconformity* functions, that provide estimates for how different a new example is from a set of old examples. In a predictive modeling scenario, nonconformity functions use the underlying model to determine how strange the relationship between the feature vector (the input) and an output value for a certain new instance is, compared to a set of previously observed instances.

It is possible to design many different nonconformity functions for a specific predictive model, and each of them will define a different conformal predictor. All of these conformal predictors will be valid, but there may be significant differences in terms of efficiency. In the extreme case, even a function that returns the same nonconformity score for all examples will be valid, but the prediction regions will be very wide.

CP was originally introduced as a transductive approach for support vector machines (Gammerman et al. 1998). Transductive CP requires learning a new model for each new test instance to be predicted, which of course may be computationally prohibitive. For this reason, inductive conformal prediction (ICP) was suggested (Vovk et al. 2006). In ICP, which is the focus of this study, only one model is induced from the training data and that model is then used for predicting all test instances. In ICP, however, the calculation of the nonconformity scores requires a separate data set (called the *calibration set*) that was not used by the algorithm when learning the model. Consequently, it becomes very important how the training data is

divided into the proper training set and the calibration set; using too few calibration instances will result in imprecise confidence values, while too few proper training instances may lead to weaker underlying models.

Looking specifically at ICP regression, there are very few published papers providing a systematic evaluation of different underlying models and nonconformity functions. As a matter of fact, until now, most studies focus on one specific underlying model, and use a very limited number of data sets, making them serve mainly as proofs-of-concept; see e.g., Papadopoulos et al. (2002); Papadopoulos and Haralambous (2011). With this in mind, there is an apparent need for larger studies, explicitly evaluating techniques for producing efficient conformal predictors. Such studies should preferably explore various learning algorithms and use a sufficiently large number of data sets to allow for statistical inference, thus making it possible to establish best practices. In this paper, we compare using random forests (Breiman 2001) as the underlying model for conformal prediction regression to existing state-of-the-art conformal regressors, which are based on artificial neural networks (ANN) and k-nearest neighbors (kNN). We investigate a number of nonconformity functions, and we specifically examine the option to use out-of-bag estimates for the necessary calibration.

In summary, the main contributions of this paper are:

- a novel method for regression conformal prediction, which utilizes random forests together with a non-conformity function that exploits out-of-bag examples as a calibration set;
- the first large-scale empirical investigation of methods for regression conformal prediction, which include state-of-the-art learning algorithms and multiple non-conformity functions that are evaluated on a large number of datasets;
- significant findings concerning the relative efficiency of different conformal predictors, which provide new evidence for what may be considered best practices for regression conformal prediction.

In the next section, we formalize the conformal prediction framework and discuss related work. In Sect. 3, we describe the proposed approach for regression conformal prediction using random forests as well as competing state-of-the-art approaches. The setup for, and the results from, the empirical investigation are presented in Sect. 4. Finally, we summarize the main conclusions from the study and outline directions for future work in Sect. 5.

2 Background

In this section, we first provide a formalization of inductive conformal prediction, which is the theoretical foundation for this paper. We then briefly discuss its relation to alternative frameworks and summarize the main related previous studies upon which our study builds.

2.1 Inductive conformal prediction

An inductive conformal classifier or regressor only needs to be trained once, using the following scheme:

1. Divide the training set $Z = \{(x_1, y_1), \ldots, (x_l, y_l)\}$ into two disjoint subsets Z^t (a proper training set) and Z^c (a calibration set):
   - $Z^t = \{(x_1, y_1), \ldots, (x_m, y_m)\}$
   - $Z^c = \{(x_{m+1}, y_{m+1}), \ldots, (x_l, y_l)\}$

2. Train the underlying model h_Z using Z^t.
3. For each calibration instance $(x_i, y_i) \in Z^c$:
   - let h_Z predict the output value for x_i so that $\hat{y}_i = h_Z(x_i)$ and
   - calculate the nonconformity score α_i using the nonconformity function.

For a novel (test) instance, the input pattern x_j is supplied to the underlying model, resulting in a prediction $\hat{y}_j$. Then a nonconformity score $\alpha_j^{\tilde{y}}$ is produced for every tentative target value $\tilde{y}$. The p-value of each tentative target $\tilde{y}$ is then calculated by comparing $\alpha_j^{\tilde{y}}$ to the nonconformity scores of the calibration set $S = \{\alpha_1, \ldots, \alpha_q\}$:

$$p(\tilde{y}) = \frac{\#\{z_i \in Z^c \mid \alpha_i \geq a_j^{\tilde{y}}\} + 1}{|Z^c| + 1}. \tag{1}$$

If $p(\tilde{y}) < \delta$, the probability for $\tilde{y}$ being the true target for x_j is smaller than δ, i.e., $\tilde{y}$ can be excluded from the prediction region, at that confidence level. In classification, $\tilde{y}$ represents a possible class label, and all possible labels are tested one at a time. In regression, we cannot consider every possible output value in that manner, so a conformal regressor will instead directly establish the prediction interval, for each test instance, given the confidence level. In regression, the nonconformity function is most often simply the absolute error, see e.g., Papadopoulos and Haralambous (2011); Papadopoulos et al. (2002, 2011):

$$\alpha_i = |y_i - \hat{y}_i|. \tag{2}$$

Then, given a significance level δ and a set of calibration scores $S = \{\alpha_1, \ldots, \alpha_q\}$, we locate the smallest $\alpha_{s(\delta)} \in S$ that satisfies the equation

$$\frac{\#\{z_i \in Z^c \mid \alpha_i < \alpha_{s(\delta)}\} + 1}{|Z^c| + 1} \geq 1 - \delta. \tag{3}$$

Since it is not possible to consider each $\tilde{y}$ in regression, it is also not possible to calculate the nonconformity scores $\alpha_j^{\tilde{y}}$ for the test instance x_j. Instead, $\alpha_{s(\delta)}$ forms a probabilistic bound for the nonconformity scores at significance level δ; that is, with probability $1 - \delta$, the nonconformity of x_j will be at most $\alpha_{s(\delta)}$. Thus, at significance δ, we can reject any label for which $\alpha_j^{\tilde{y}} > \alpha_{s(\delta)}$, and must conversely include all labels for which $\alpha_j^{\tilde{y}} \leq \alpha_{s(\delta)}$. Using (2), $\alpha_j^{\tilde{y}} = \alpha_{s(\delta)}$ exactly when $|y - \tilde{y}| = \alpha_{s(\delta)}$, hence, by formulating the prediction region as

$$\hat{Y}_j^{\delta} = \hat{y}_j \pm \alpha_{s(\delta)}, \tag{4}$$

where $s(\delta)$ is found from (3) above, $\hat{Y}_j^{\delta}$ will cover the true output y_j with probability $1 - \delta$. It must be noted that when using (2) and (4), the conformal regressor will, for any specific significance level δ, always produce prediction intervals of the same size for every x_j; i.e., it does not consider the difficulty of a certain instance x_j in order to provide as informative predictions as possible, which often is a key motivation for using conformal prediction in the first place. It is, however, possible to employ *normalized* nonconformity functions, where the absolute error is scaled using the expected accuracy of the underlying model; see e.g., Papadopoulos and Haralambous (2011); Papadopoulos et al. (2011). The motivation for this, from a conformal prediction standpoint, is that if two instances have identical conformity scores using (2), but the first is expected to be more accurate than the second, then the second is actually stranger (more nonconforming) than the first. Using a normalized nonconformity function, the resulting prediction intervals will be smaller for instances that are deemed

"easy" and larger for "harder" instances. When using a normalized nonconformity function, nonconformity scores are calculated using:

$$\alpha_i = \frac{|y_i - \hat{y}_i|}{\sigma_i}, \tag{5}$$

where σ_i is an estimate of the accuracy of the underlying model for $\hat{y}_i$. Naturally, there are several ways to estimate the accuracy; one suggestion is to train another model for predicting the errors; see e.g., Papadopoulos and Haralambous (2011). Other approaches use properties of the underlying model; see e.g., Papadopoulos et al. (2011). With normalized nonconformity functions, the prediction interval for $\hat{Y}_j^\delta$ is:

$$\hat{Y}_j^\delta = \hat{y}_j \pm \alpha_{s(\delta)}\sigma_j, \tag{6}$$

where σ_j is an estimate of the accuracy of the underlying model, for that instance.

2.2 Related work

As mentioned in the introduction, there are other machine learning frameworks that provide some sort of guarantee of the prediction error. Specifically, PAC-learning (Valiant 1984) will provide upper bounds on the probability of its error with respect to some confidence level. PAC theory only assumes that the instances are generated independently by some completely unknown distribution, but for the resulting bounds to be interesting in practice, the data set must be quite clean. Unfortunately, this is rarely the case for real-world data, which will lead to very loose bounds, see e.g., Nouretdinov et al. (2001), where the crudeness of PAC theory is demonstrated. In addition, the PAC bounds are for the overall error and not for individual predictions. The Bayesian framework can, on the other hand, be used to complement individual predictions with probabilistic measures of their quality. These measures are, however, based on some *a priori* assumption about the underlying distribution. When the assumed prior is violated, there is no guarantee that the resulting intervals produced by the Bayesian methods actually contain the true target as often as indicated by the confidence level, i.e., the resulting predictions are not valid. In Papadopoulos et al. (2011), CP is compared to the popular Bayesian method called Gaussian Processes (GP) (Rasmussen and Christopher 2005). The results show that when the (artificial) data set satisfied the GP prior, the intervals produced by GP-regression were valid, and slightly tighter than the corresponding intervals produced by CP. On a number of real-world data sets, however, the predictive regions produced by GP-regression were no longer valid, i.e., they may become misleading when the correct prior is not known.

The CP framework has been applied to classification using several popular learning algorithms, such as ANNs (Papadopoulos 2008), kNN (Nguyen and Luo 2012), SVMs (Devetyarov and Nouretdinov 2010; Makili et al. 2011), decision trees (Johansson et al. 2013a), random forests (Bhattacharyya 2011; Devetyarov and Nouretdinov 2010) and evolutionary algorithms (Johansson et al. 2013b; Lambrou et al. 2011). Although we in this study consider regression tasks, there is some overlap with previous studies on classification when it comes to design choices. Specifically, in Johansson et al. (2013a), the underlying learning algorithm is also decision trees. However, in the previous study, the focus was on how properties of the algorithm, e.g. split evaluation metric, pruning and the smoothening function, affect the efficiency of classification trees. In this paper, we instead study forests of regression trees, and investigate both standard and normalized nonconformity functions in this context. Moreover, the use of out-of-bag estimates for the calibration was suggested for random forests in Devetyarov and Nouretdinov (2010), and was also used for bagged ANNs in Löfström et al.

(2013). None of these studies, however, evaluate efficiency in a systematic way while considering different underlying models and nonconformity functions. In particular, no normalized nonconformity functions were evaluated in these studies.

There are also a number of studies on conformal prediction for regression, using, for instance, ridge regression (Papadopoulos et al. 2002) and ANNs (Papadopoulos and Haralambous 2010). Two interesting and fairly recent studies do in fact evaluate normalized nonconformity functions for ANNs (Papadopoulos and Haralambous 2011) and k-Nearest Neighbors (Papadopoulos et al. 2011). Unfortunately, both studies use very few data sets, thus precluding statistical analysis. Despite these shortcomings, the suggested approaches must be regarded as state-of-the-art for ICP regression, making them natural benchmarks to compare our proposed methods against.

Conformal prediction has also been successfully used in a number of applications where confidence in the predictions is of concern, including prediction of space weather parameters (Papadopoulos and Haralambous 2011), estimation of software project effort (Papadopoulos et al. 2009b), early diagnostics of ovarian and breast cancers (Devetyarov et al. 2012) and diagnosis of acute abdominal pain (Papadopoulos et al. 2009a).

3 Methods

In this section, we first describe the proposed method for utilizing random forests for regression conformal prediction, including some variants, and then describe the competing state-of-the-art approaches.

3.1 Regression conformal prediction using random forests

A random forest (Breiman 2001) is a set of decision trees (Breiman et al. 1984; Quinlan 1986), where each tree is generated in a specific way to introduce diversity among the trees, and where predictions of the forest are formed by voting. A decision tree is a tree-structured (directed, acyclic and connected) graph, where each internal (non-leaf) node is labeled with a test on some attribute, with one arc leading to a unique (child) node for each possible outcome of the test, and where the leaf nodes of the tree are labeled with values to be predicted. If the predicted values are (categorical) class labels, the decision tree is called a classification tree, while if the predicted values are numeric, the tree is called a regression tree. When using a decision tree to predict a value for an example, starting at the root node, the test at the current internal node is performed and the arc corresponding to the outcome of the test is followed, until a leaf node is reached, for which the corresponding predicted value is returned. For a forest of classification trees, the predicted value is typically formed by selecting the majority among the predictions of the individual trees, while for a forest of regression trees, the resulting prediction is typically formed by taking the average of the individual predictions.

The standard procedure to generate a decision tree is to employ a recursive partitioning, or divide-and-conquer, strategy, starting with all training examples at the root node of the tree, and then evaluating all available tests to partition the examples, choosing the one that maximizes some evaluation metric, e.g., information gain for classification trees or variance for regression trees, labeling the current node with this test, partitioning the examples according to the outcome of the test and continuing building the tree recursively with each resulting subset, until some termination criterion is met, e.g., all examples in the subset have the same value on the target attribute, and forming a value to predict from the examples in the subset.

In order to introduce the necessary diversity among the trees in a random forest, each tree is trained on a *bootstrap replicate* of the original training set (Breiman 1996), i.e., a new training (multi-)set, or bag, of the same size as the original set is formed by randomly selecting examples with replacement from the original set. This means that some of the original examples may be duplicated in the bootstrap replicate, while other examples are excluded. The latter ones, for a specific tree, are said to be *out-of-bag* for that tree. To further increase diversity, each tree in the forest is created using the random subspace method (Ho 1998), i.e., only a randomly selected subset of all available attributes are evaluated when choosing the split at each internal node during the construction of the decision tree.

In this study, the implementation of the random forest algorithm from the MatLab [1] statistics toolbox, called TreeBagger, was used. The parameters were set to the default values for regression trees, i.e., the number of attributes to evaluate at each internal node was set to one third of the total number of attributes and mean square error was used as the split criterion.

Given that random forests are frequently observed to result in state-of-the-art predictive performance, see e.g., Caruana and Niculescu-Mizil (2006), random forest models can be expected to be more accurate than the underlying models that are currently used in regression conformal predictors, i.e., kNN and ANN models. It is, however, not obvious whether or not the use of random forests will result in smaller prediction intervals, when the models are used as the basis for CP. Another important question is whether or not anything could be gained from using out-of-bag estimates for the calibration, something which is an option for random forests, but not for the previous model types, which have to resort to using a separate calibration set.

In this study, we investigate nonconformity functions that are based on absolute errors (2). The first two nonconformity functions that will be considered for random forests use no normalization, i.e., the intervals are produced using (4). The first approach, called RFi, employs standard ICP, i.e., a separate calibration set is used. In the second approach, called RFo, out-of-bag instances are instead used for the calibration. This, of course, makes it possible to use all training instances for both the training and the calibration. More specifically, when producing the nonconformity score for a calibration instance z_i, the ensemble used for producing the prediction $\hat{y}_i$ consists of all trees that were not trained using z_i, i.e., z_i was out-of-bag for those trees.

It should be noted that when using out-of-bag instances instead of a separate calibration set, the actual underlying model, i.e., the random forest, is no longer used when calculating the nonconformity scores and p-values. In fact, various subsets of the forest are used for the out-of-bag-instances, but the entire forest is used for the test instances. In other words, the nonconformity functions applied to the calibration and test instances are defined differently as

$$\alpha_{calibration} = |y - h\theta(x)| \tag{7}$$

$$\alpha_{test} = |y - h(x)|\,, \tag{8}$$

where θ is a random factor determining the subset of trees for which x is out-of-bag. In general, the use of different nonconformity functions could clearly cause the resulting conformal predictor to become invalid, i.e., the probability of excluding the true target value would no longer be bounded by the provided confidence level. However, we argue that the conformal predictor in our particular setting, i.e., when using out-of-bag estimates for the calibration, must be valid. In principle, the same random component may also be used when predicting

[1] www.mathworks.com.

the target value for the test instance (by only considering a random subset of the forest when predicting the target of the test instance), and in that case the same nonconformity function (7) would obviously be used for all instances, hence not violating the assumptions underlying the ICP framework. When instead using the whole forest for the test instance, as proposed here, one would expect the predicted values to be closer to the true target, than when using a random subset of the trees. In fact, it is well-known that out-of-bag error estimates tend to overestimate the actual error made by a random forest, simply because a larger forest is normally a stronger model. Not until the random forest is so large that the randomized sub-ensembles will be as accurate as the entire forest, is this bias eliminated. Consequently, the expected nonconformity of a test instance is less than (or for a very large forest equal to) the expected nonconformity of a calibration instance, i.e., the probability of including nonconforming targets in the prediction region is unchanged or increased when using the whole forest. Hence, rather than increasing the risk for generating an invalid conformal predictor, one would expect the conformal predictor using out-of-bag instances to be conservative. Therefore, the proposed setup should be, if anything, less efficient than if the whole forest was used together with additional calibration instances. Naturally, the validity will be investigated in the experimentation in order to support this reasoning empirically.

We also investigate three normalized nonconformity functions, i.e., the prediction regions may vary for different test instances. RFia and RFoa both use an additional (linear) ANN to predict the logarithm of the error of the underlying model, for each instance. RFia is identical to the procedure used in Papadopoulos and Haralambous (2011) and Papadopoulos and Haralambous (2010), but of course uses a random forest as the underlying model instead of an ANN. The resulting nonconformity function is:

$$\alpha_i = \frac{|y_i - \hat{y}_i|}{exp(\mu_i) + \beta}, \tag{9}$$

where μ_i is the prediction of the value $ln(|y_i - \hat{y}_i|)$ produced by the linear ANN, and β is a parameter, used to control the sensitivity of the nonconformity measure. Naturally, this ANN was trained on all pairs $(x_j, ln(|y_j - \hat{y}_j|)$, from the proper training set. Using this nonconformity function, the prediction intervals become:

$$\hat{Y}_j^{\delta} = \hat{y}_j \pm \alpha_{s(\delta)}(exp(\mu_j) + \beta)\,. \tag{10}$$

The only difference between RFia and the novel setup RFoa is that RFia uses a separate calibration set, while RFoa uses the out-of-bag instances for the calibration, i.e., the additional ANN is trained using the logarithm of the out-of-bag errors as targets. RFok, finally, is another novel setup, which instead of employing an additional ANN for predicting the logarithm of the error, considers the average out-of-bag error (normalized with the Euclidean distance) for the k closest instances. That is, RFok is based on the same Eq. (9), but here, μ_i is defined as the logarithm of the average out-of-bag error of the k nearest neighbors. The motivation for this novel, and quite straightforward nonconformity function, is that if neighboring instances have small out-of-bag errors, the prediction for the new instance should be accurate, i.e., that instance should be considered as relatively easy. The exact number of neighbors to use is optimized (between 1 and 45) for each fold based on the average interval size of the resulting conformal regressor. This fitting is similar to when the ANN is trained to learn the out-of-bag errors. Naturally, since both RFoa and RFok utilize the out-of-bag estimates for the calibration, they make it possible to use all available data as a proper training set for the random forest.

Since the normalization functions used in RFoa and RFok depend on the out-of-bag errors of the calibration instances, one may again raise concerns on the validity of the corresponding

conformal regressors. Starting with RFok, we claim that this normalization is unbiased. More specifically, the difficulty of any instance (calibration or test) is estimated in the same manner, using its k closest calibration set neighbors, not counting a calibration instance as its own closest neighbor. Hence, for both test and calibration instances, the difficulty estimate is based on a set of examples that does not include the instance to which the estimate applies. Consequently, there is no reason to suspect that the estimated difficulty of a test instance is any less accurate than that of a calibration instance. The error rate of RFok is thus not expected to be affected by the normalization function used, and RFok is therefore expected to keep the (slightly conservative) validity of the RFo nonconformity function it is based on. In RFoa, on the other hand, the difficulty-estimating ANN model has been trained on the out-of-bag error of all calibration instances, i.e., when predicting the difficulty of some calibration instance, that particular instance will have been used in the training of the ANN, whereas the same does not apply to any test instance. So, in this case, there is indeed a bias towards the calibration set.

Consequently, for RFoa there are two forces working in opposite directions; the inherent conservatism in using out-of-bag estimates and the bias towards the calibration set when estimating the difficulty of an instance. When the latter bias is small, e.g., if the ANN is relatively weak, the resulting error rate will most likely be smaller than the confidence level, but for a larger bias, the error rate may actually be higher than the confidence level. With this in mind, it is important to recognize that RFoa is the only setup evaluated for which there is a known risk that validity is not guaranteed. Again, the empirical investigation will study how the error rate is affected by these nonconformity functions in practice.

3.2 Competing approaches

In the empirical evaluation, we compare the different variants of our suggested method to the state-of-the-art techniques from Papadopoulos and Haralambous (2011) and Papadopoulos et al. (2011). In both these papers, ICP methods were used, i.e., separate calibration sets were required. The first competing method, suggested and described in detail in Papadopoulos and Haralambous (2011), uses an ANN as the underlying model. In the most basic format (here referred to as ANN), it uses the standard nonconformity function (2) and produces intervals using (4). When using a normalized nonconformity function, the method, which is here referred to as ANNa, uses a linear ANN to predict the logarithm of the errors, and produces intervals using (9) and (10).

The second competing method is based on distance-weighted k-nearest neighbor regressors, and is suggested and described in Papadopoulos et al. (2011). In the basic format, this method (referred to as kNN) also uses the standard way of calculating nonconformity scores (2) and prediction intervals (4). In Papadopoulos et al. (2011), the authors evaluate a number of novel normalized nonconformity functions. The most efficient (here called kNNc) combines two different aspects of kNN, in order to produce as good estimates of the accuracy as possible. More specifically, the prediction from a kNN regressor is deemed to be more accurate, for a specific instance, if (i) the k nearest neighbors are close to the current test instance and (ii) the k nearest neighbors agree in their predictions. The resulting nonconformity function is

$$\alpha_i = \frac{|y_i - \hat{y}_i|}{exp(\gamma\lambda_i) + exp(\rho\xi_i)}, \tag{11}$$

where λ and ξ are the measures of accuracy (difficulty) while γ and ρ are parameters controlling the sensitivity of each measure. Consequently, the prediction intervals are calculated using

$$\hat{Y}_j^{\delta} = \hat{y}_j \pm \alpha_{s(\delta)}(exp(\gamma\lambda_j) + exp(\rho\xi_j)) \,. \tag{12}$$

It must be noted that an internal normalization of the measures is used to make sure that the two measures are of the same magnitude and robust over all data sets. For a detailed discussion on these and other difficulty estimators, see Papadopoulos et al. (2011).

4 Empirical evaluation

In this section, we first describe the experimental setup, i.e., what algorithms, data sets and performance metrics have been chosen, and then report the results from the experiment.

4.1 Experimental setup

In the first (main) experiment, the competing methods were re-implemented, and a large-scale study, using 33 publicly available data sets was performed. The considered data sets are small to medium sized; ranging from approximately 500 to 10, 000 instances. All but one data set are from the UCI Bache and Lichman (2013), Delve Rasmussen et al. (1996) or KEEL Alcalá-Fdez et al. (2011) repositories. The data sets are described in Table 1, where *#inst.* is the number of instances, *#attrib.* is the number of input attributes and *#calInst*, is the number of instances used for calibration in the standard ICP settings.

In the evaluation, we look at standard and normalized nonconformity functions separately. Naturally, the normalized nonconformity functions are the most important, since they provide prediction intervals of different sizes. In the second experiment, we employed the exact same settings as in the previous studies, including using only a handful of data sets, and compare our results directly to the published results.

In Experiment 1, a 10×10-fold cross-validation scheme was used. The number of calibration instances was set to

$$q = 100 \times \left\lfloor \frac{|Z|}{400} \right\rfloor - 1, \tag{13}$$

where Z is the full training set, i.e., starting at 99 calibration instances for data sets with 400–799 examples, and adding 100 calibration instances for every additional 400 examples in the full training set. Before the experimentation, all target values were normalized to [0, 1], in order to obtain more readable efficiency comparisons across data sets. With this scaling, the size of a prediction interval, of course, expresses the fraction of the target range covered by the interval.

Regarding parameter values, we elected to use identical settings over all data sets and, when applicable, methods. Specifically, all random forests consisted of 500 random trees. For kNN regressors, k was set to 25, since some preliminary experiments showed that this actually produced higher efficiency than selecting different k-values based on internal cross-validation results. Similarly, all ANNs had exactly 20 hidden units. The sensitivity parameters had to be adjusted based on the much smaller normalized target ranges. Again, some preliminary experiments showed that the exact values were not vital, so the following values were used in Experiment 1: $\beta = 0.01$ and $\gamma = \rho = 1.0$.

Two things were measured for each method and data set in the experiments; the *error rate*, i.e., the fraction of target values in the test set that fall outside the predicted regions, and the *efficiency*, i.e., the size of the predicted intervals. For valid conformal predictors, the error rates should not (in the long run) exceed the chosen confidence threshold. Hence, by investigating the error rates, we may confirm (or reject) that a certain conformal predictor actually is

valid. Note that this is here considered to be a binary property, i.e., we do not consider one method to be more valid than another. Given that we have a set of valid regression conformal predictors, the perhaps most interesting aspect to compare is the size of the predicted regions, as this directly corresponds to how informative these regions are. Such a comparison could be done in different ways, e.g., comparing extreme values, but we have opted for comparing the average sizes over all prediction regions. In fact, we report the median value from the ten runs of ten-fold cross-validation.

In order to be able to do a direct comparison with published results, we used the same settings for our methods in Experiment 2 as originally employed for the specific data sets,

Table 1 Data set characteristics

Name	#inst.	#attrib.	#calInst.	Origin
abalone	4177	8	899	UCI
anacalt	566	7	99	KEEL
bank8fh	8192	8	1799	Delve
bank8fm	8192	8	1799	Delve
bank8nh	8192	8	1799	Delve
bank8nm	8192	8	1799	Delve
boston	506	13	99	UCI
comp	8192	12	1799	Delve
concreate	992	8	199	UCI
cooling	768	8	99	UCI
deltaA	7129	5	1599	KEEL
deltaE	9517	6	2099	KEEL
friedm	1200	5	199	KEEL
heating	768	8	99	UCI
istanbul	536	7	99	UCI
kin8fh	8192	8	1799	Delve
kin8fm	8192	8	1799	Delve
kin8nh	8192	8	1799	Delve
kin8nm	8192	8	1799	Delve
laser	993	4	199	KEEL
mg	1385	6	299	Flake and Lawrence (2002)
mortage	1048	15	199	KEEL
plastic	1055	2	199	KEEL
puma8fh	8192	8	1799	Delve
puma8fm	8192	8	1799	Delve
puma8nh	8192	8	1799	Delve
puma8nm	8192	8	1799	Delve
quakes	2178	2	399	KEEL
stock	950	9	199	KEEL
treasury	1048	15	199	KEEL
wineRed	1359	11	299	UCI
wineWhite	3961	11	799	UCI
wizmir	1460	2	299	KEEL

Table 2 Error rates for standard nonconformity functions

	Error rates											
Confidence	90 %				95 %				99 %			
Technique	ANN	kNN	RFi	RFo	ANN	kNN	RFi	RFo	ANN	kNN	RFi	RFo
abalone	.101	.101	.101	.100	.050	.051	.051	.050	.010	.010	.009	.010
anacalt	.095	.098	.100	.095	.051	.049	.050	.050	.012	.010	.009	.009
bank8fh	.100	.100	.099	.099	.050	.050	.050	.049	.010	.010	.010	.010
bank8fm	.100	.100	.101	.098	.050	.051	.050	.049	.010	.010	.010	.010
bank8nh	.099	.099	.100	.100	.050	.050	.050	.050	.010	.010	.010	.010
bank8nm	.101	.100	.100	.100	.050	.050	.050	.049	.010	.010	.010	.010
boston	.100	.104	.103	.097	.048	.046	.049	.047	.010	.008	.009	.008
comp	.101	.100	.100	.099	.050	.050	.050	.049	.010	.010	.010	.010
concreate	.102	.102	.101	.097	.053	.052	.052	.047	.012	.010	.010	.008
cooling	.094	.096	.099	.101	.047	.049	.048	.047	.008	.010	.010	.008
deltaA	.100	.100	.100	.100	.049	.049	.050	.050	.010	.010	.009	.010
deltaE	.100	.099	.100	.100	.050	.049	.049	.050	.010	.010	.010	.010
friedm	.100	.103	.104	.102	.049	.052	.052	.049	.010	.011	.010	.009
heating	.106	.101	.101	.098	.050	.050	.049	.049	.012	.009	.013	.007
istanbul	.096	.093	.097	.100	.051	.047	.048	.049	.010	.011	.010	.009
kin8fh	.099	.100	.100	.098	.050	.049	.050	.049	.010	.010	.010	.010
kin8fm	.101	.101	.100	.097	.050	.051	.050	.049	.011	.010	.010	.010
kin8nh	.099	.099	.099	.099	.050	.050	.051	.050	.010	.010	.010	.010
kin8nm	.100	.100	.100	.098	.050	.051	.051	.049	.009	.010	.010	.010
laser	.098	.102	.102	.098	.046	.048	.053	.050	.010	.009	.010	.008
mg	.098	.096	.098	.097	.050	.050	.048	.049	.009	.010	.009	.009
mortage	.103	.100	.101	.098	.051	.052	.054	.048	.011	.011	.011	.009
plastic	.104	.103	.099	.101	.049	.050	.050	.050	.010	.008	.009	.010
puma8fh	.101	.102	.101	.100	.051	.051	.051	.050	.010	.010	.010	.010
puma8fm	.100	.100	.100	.099	.050	.049	.050	.050	.010	.010	.010	.010
puma8nh	.100	.101	.100	.100	.050	.051	.050	.050	.010	.010	.010	.010
puma8nm	.102	.098	.100	.097	.051	.050	.050	.048	.010	.010	.010	.010
quakes	.102	.102	.104	.100	.053	.053	.053	.049	.011	.011	.010	.010
stock	.105	.101	.101	.095	.052	.051	.050	.047	.011	.011	.010	.009
treasury	.103	.103	.105	.097	.052	.052	.053	.048	.012	.010	.009	.009
wineRed	.100	.099	.097	.100	.049	.050	.050	.049	.009	.010	.010	.010
wineWhite	.100	.101	.100	.099	.051	.050	.050	.050	.010	.010	.010	.010
wizmir	.098	.101	.097	.099	.051	.050	.049	.048	.011	.010	.010	.010
Mean	**.100**	**.100**	**.100**	**.099**	**.050**	**.050**	**.050**	**.049**	**.010**	**.010**	**.010**	**.009**

with regard to number of folds and number of calibration instances. In addition, in this experiment, the targets were not normalized. It may be noted that parameters like k in kNN and the number of hidden neurons in the ANNs, in the original studies, were optimized based on accuracy results using internal cross-validation. All sensitivity parameters (β, γ and ρ) were, however, despite the fact that the importance of the parameters will be heavily affected

Table 3 Efficiency for standard nonconformity functions

	Median interval sizes											
Confidence	90 %				95 %				99 %			
Technique	ANN	kNN	RFi	RFo	ANN	kNN	RFi	RFo	ANN	kNN	RFi	RFo
abalone	.238	.239	.233	.235	.320	.337	.317	.320	.541	.590	.550	.540
anacalt	.173	.384	.183	.178	.281	.821	.390	.378	.774	1.379	.748	.644
bank8fh	.290	.339	.301	.301	.373	.425	.382	.381	.544	.590	.538	.538
bank8fm	.132	.226	.138	.136	.164	.279	.174	.171	.234	.374	.267	.259
bank8nh	.321	.349	.327	.325	.441	.488	.448	.446	.786	.829	.775	.778
bank8nm	.148	.184	.151	.149	.206	.286	.216	.212	.358	.519	.404	.389
boston	.239	.299	.214	.203	.329	.483	.325	.311	.658	1.114	.712	.660
comp	.118	.105	.089	.088	.151	.140	.117	.116	.233	.272	.196	.195
concreate	.285	.384	.228	.215	.352	.465	.283	.263	.523	.687	.478	.499
cooling	.269	.266	.181	.179	.335	.391	.225	.222	.444	.570	.308	.300
deltaA	.125	.123	.116	.115	.164	.162	.154	.153	.265	.282	.264	.258
deltaE	.175	.175	.172	.172	.215	.220	.214	.213	.318	.330	.316	.315
friedm	.164	.228	.231	.228	.202	.286	.279	.277	.284	.391	.380	.378
heating	.205	.295	.073	.070	.263	.349	.099	.095	.349	.443	.231	.215
istanbul	.280	.266	.268	.265	.344	.337	.325	.313	.555	.509	.491	.494
kin8fh	.231	.243	.243	.242	.277	.294	.297	.294	.379	.408	.402	.403
kin8fm	.101	.134	.147	.143	.122	.168	.182	.177	.169	.248	.262	.254
kin8nh	.403	.393	.415	.411	.481	.464	.496	.489	.627	.607	.640	.637
kin8nm	.296	.292	.342	.332	.365	.353	.410	.401	.512	.475	.559	.546
laser	.080	.077	.059	.054	.123	.149	.104	.095	.397	.607	.381	.337
mg	.364	.338	.290	.267	.471	.454	.402	.374	.751	.679	.691	.657
mortage	.033	.054	.032	.029	.043	.077	.050	.045	.079	.148	.109	.097
plastic	.494	.514	.515	.509	.573	.598	.596	.584	.717	.763	.756	.735
puma8fh	.464	.486	.469	.471	.566	.577	.569	.574	.753	.756	.762	.766
puma8fm	.209	.275	.210	.210	.254	.328	.255	.254	.349	.434	.350	.349
puma8nh	.452	.496	.440	.438	.559	.597	.542	.543	.779	.816	.744	.748
puma8nm	.209	.366	.187	.182	.258	.443	.231	.225	.368	.623	.333	.321
quakes	.491	.528	.520	.522	.692	.694	.692	.704	1.015	1.040	1.052	1.044
stock	.123	.118	.101	.093	.150	.143	.130	.119	.217	.211	.222	.198
treasury	.041	.046	.034	.033	.057	.067	.055	.053	.113	.128	.121	.109
wineRed	.434	.449	.420	.416	.552	.570	.534	.528	.785	.778	.763	.750
wineWhite	.398	.387	.372	.370	.495	.488	.469	.466	.713	.725	.708	.700
wizmir	.067	.142	.069	.067	.084	.177	.086	.083	.155	.289	.162	.146
Mean	**.244**	**.279**	**.236**	**.232**	**.311**	**.367**	**.304**	**.299**	**.477**	**.564**	**.475**	**.462**
Mean rank	**2.45**	**3.36**	**2.52**	**1.67**	**2.33**	**3.52**	**2.48**	**1.67**	**2.27**	**3.45**	**2.55**	**1.73**

by the actual range of the target values, somewhat ad hoc set to 0.5. Consequently, we too set $\beta = 0.5$ for all our methods in Experiment 2.

All experimentation was performed in MatLab, in particular using the Neural network and the Statistics toolboxes.

Table 4 Standard nonconformity functions

	90 %	95 %	99 %
RFo versus kNN	**6E−07**	**1E−07**	**9E−07**
RFo versus ANN	**0.026**	0.057	0.058
RFo versus RFi	**0.026**	0.051	0.058
RFi versus kNN	**0.013**	**0.002**	**0.007**
RFi versus ANN	0.775	0.634	0.849
ANN versus kNN	**0.008**	**0.001**	**0.006**

Adjusted p values. Numbers in bold are significant differences at $\alpha = 0.05$

4.2 Experimental results

Table 2 demonstrates validity for the methods utilizing standard nonconformity functions. Looking at the error rates, i.e., the fraction of test instances for which the true target value falls outside the predicted region, it is reassuring to see that the empirical results for each and every data set is very close to the predetermined confidence levels. In addition, it can be noted that RFo tends to be slightly conservative, which supports the reasoning about validity in Sect. 3.1.

Looking at the interval sizes tabulated in Table 3, while remembering that the output was normalized so that an interval size of 1.0 would cover the entire range of the target values, it can be seen from the averaged values that the methods at the 90 % confidence level returned valid prediction intervals covering, approximately, 25 % of the range. The corresponding average values for the 95 and 99 % confidence levels are (approximately) 30 and 50 %, respectively. Clearly, these valid prediction intervals must be considered informative.

In order to compare the efficiency of the five different techniques, and to find out if there are any statistically significant differences, we used the recommended procedure in García and Herrera (2008) and performed a Friedman test Friedman (1937), followed by Bergmann–Hommel's dynamic procedure Bergmann and Hommel (1988) to establish all pairwise differences. Table 4 shows the adjusted p values. The most important result is that RFo is either significantly or substantially more efficient than all three competing methods, at the different confidence levels. Specifically, it should be noted that RFo outperformed RFi, clearly showing that the use of out-of-bag instances for the calibration is beneficial.

Table 5 demonstrates the validity for the methods utilizing normalized nonconformity functions. Again we see that all methods, including RFoa, produced valid and well-calibrated conformal predictors. Actually, when using 10×10-fold cross-validation, the empirical error rates for most individual data sets are very close to the confidence level. Clearly, these results support our argumentation above that the suggested setup RFok will produce valid (possibly slightly conservative) conformal regressors. Although RFoa was argued above to be associated with a risk of producing non-valid predictions due to a bias in the difficulty estimation function, using the specific settings and parameter values employed here, this bias turned out to be compensated for by the conservative out-of-bag error estimates, thus resulting in empirical error rates below the confidence threshold even for this nonconformity function.

Table 6 shows the interval widths for the normalized nonconformity functions. Comparing these to the results in Table 3, it is obvious that the prediction intervals here are much smaller, i.e., applying normalized nonconformity functions does not only provide tuned prediction intervals for each specific test instance, but the resulting intervals are also substantially tighter on average. Looking at individual methods, the mean ranks identify three groups, which are

Table 5 Error rates for normalized nonconformity functions

	Error rates														
Confidence	90 %					95 %					99 %				
Techniques	ANNa	kNNc	RFia	RFoa	RFok	ANNa	kNNc	RFia	RFoa	RFok	ANNa	kNNc	RFia	RFoa	RFok
abalone	.100	.103	.102	.101	.103	.050	.050	.051	.050	.052	.009	.010	.010	.010	.012
anacalt	.101	.097	.098	.100	.103	.052	.048	.050	.050	.051	.012	.011	.009	.010	.012
bank8fh	.101	.099	.100	.100	.101	.050	.050	.050	.049	.051	.010	.010	.009	.010	.011
bank8fm	.100	.100	.101	.097	.099	.051	.050	.051	.050	.050	.010	.010	.010	.010	.010
bank8nh	.099	.099	.099	.099	.101	.050	.050	.050	.050	.051	.010	.010	.010	.010	.011
bank8nm	.101	.101	.100	.099	.098	.051	.050	.050	.049	.048	.010	.010	.010	.010	.010
boston	.100	.099	.104	.103	.100	.048	.055	.047	.050	.051	.011	.010	.008	.009	.011
comp	.100	.099	.099	.099	.100	.050	.050	.049	.049	.050	.011	.009	.010	.010	.011
concreate	.102	.103	.099	.097	.094	.052	.052	.050	.047	.047	.012	.010	.011	.009	.009
cooling	.099	.102	.099	.098	.095	.045	.051	.047	.053	.048	.008	.010	.010	.006	.008
deltaA	.100	.100	.100	.100	.101	.050	.050	.050	.049	.051	.010	.010	.010	.010	.010
deltaE	.099	.100	.100	.100	.101	.049	.050	.050	.050	.051	.010	.010	.010	.010	.010
friedm	.100	.103	.101	.101	.098	.048	.053	.050	.050	.048	.009	.010	.011	.009	.007
heating	.103	.098	.102	.098	.086	.051	.051	.048	.047	.038	.013	.011	.011	.008	.006
istanbul	.097	.098	.098	.106	.110	.050	.047	.048	.053	.054	.009	.010	.009	.011	.010
kin8fh	.099	.100	.100	.099	.100	.050	.049	.049	.049	.049	.010	.010	.010	.010	.010
kin8fm	.100	.100	.100	.098	.093	.051	.051	.050	.049	.046	.010	.010	.010	.009	.009
kin8nh	.100	.099	.100	.099	.099	.050	.050	.050	.050	.049	.010	.010	.010	.010	.010
kin8nm	.100	.099	.101	.100	.094	.050	.050	.051	.049	.046	.010	.010	.010	.010	.009
laser	.100	.099	.101	.097	.085	.048	.050	.052	.049	.043	.010	.011	.011	.009	.010
mg	.098	.099	.099	.098	.093	.048	.049	.049	.050	.047	.010	.009	.009	.010	.014
mortage	.103	.099	.103	.099	.088	.053	.050	.053	.050	.041	.010	.009	.010	.009	.006
plastic	.102	.102	.097	.101	.095	.049	.050	.050	.052	.044	.009	.010	.010	.009	.013
puma8fh	.101	.102	.101	.099	.101	.051	.051	.052	.050	.051	.010	.010	.010	.010	.011
puma8fm	.101	.100	.099	.099	.100	.050	.049	.049	.049	.050	.010	.010	.010	.010	.010
puma8nh	.101	.101	.100	.100	.101	.050	.050	.051	.050	.051	.010	.010	.010	.010	.010
puma8nm	.102	.100	.100	.098	.098	.051	.049	.050	.049	.050	.009	.010	.009	.009	.010
quakes	.102	.101	.103	.099	.104	.052	.051	.052	.049	.053	.012	.010	.010	.010	.013
stock	.105	.101	.102	.096	.089	.051	.049	.051	.048	.043	.011	.010	.009	.009	.006
treasury	.102	.104	.104	.096	.089	.053	.052	.051	.050	.045	.011	.011	.011	.010	.010
wineRed	.098	.097	.101	.103	.109	.051	.050	.050	.053	.053	.010	.011	.009	.010	.011
wineWhite	.100	.100	.100	.098	.102	.049	.050	.051	.050	.050	.011	.010	.010	.010	.011
wizmir	.099	.100	.098	.100	.101	.049	.050	.049	.049	.050	.010	.010	.010	.010	.011
Mean	**.101**	**.100**	**.100**	**.099**	**.098**	**.050**	**.050**	**.050**	**.050**	**.049**	**.010**	**.010**	**.010**	**.010**	**.010**

the same for all confidence levels; RFok is the most efficient method, followed by RFoa and then the other three methods. Clearly, this is a very strong result in favor of using random forests with out-of-bag examples for calibration.

Studying the adjusted p values in Table 7, we see that RFok is indeed significantly more efficient (for $\alpha = 0.05$) than all other methods, with the exception of RFoa, on all three

Table 6 Efficiency for normalized nonconformity functions

	Median interval sizes														
Confidence	90 %					95 %					99 %				
Techniques	ANNa	kNNc	RFia	RFoa	RFok	ANNa	kNNc	RFia	RFoa	RFok	ANNa	kNNc	RFia	RFoa	RFok
abalone	.204	.205	.199	.197	.197	.267	.272	.257	.251	.256	.442	.491	.433	.419	.434
anacalt	.128	.296	.124	.109	.058	.200	.514	.251	.217	.109	.611	.985	.578	.433	.283
bank8fh	.251	.291	.260	.256	.266	.323	.374	.329	.325	.334	.536	.575	.524	.516	.539
bank8fm	.120	.180	.111	.107	.113	.144	.216	.135	.130	.135	.196	.284	.185	.178	.175
bank8nh	.269	.293	.278	.272	.284	.382	.420	.382	.376	.393	.765	.796	.749	.753	.753
bank8nm	.114	.074	.094	.086	.085	.143	.102	.124	.114	.110	.220	.172	.199	.180	.167
boston	.232	.237	.207	.192	.179	.313	.307	.282	.253	.233	.616	.665	.656	.514	.414
comp	.109	.081	.080	.079	.077	.136	.104	.102	.099	.098	.202	.176	.159	.155	.154
concreate	.254	.352	.218	.200	.191	.316	.415	.266	.243	.237	.483	.559	.414	.401	.402
cooling	.173	.183	.088	.082	.070	.204	.214	.113	.101	.081	.289	.289	.161	.148	.103
deltaA	.120	.100	.111	.110	.099	.153	.132	.144	.142	.127	.244	.201	.239	.231	.193
deltaE	.174	.164	.171	.171	.167	.216	.212	.214	.213	.213	.316	.322	.318	.315	.308
friedm	.166	.222	.231	.227	.212	.202	.273	.280	.274	.256	.278	.383	.382	.371	.325
heating	.130	.138	.065	.062	.055	.153	.158	.084	.078	.067	.208	.210	.182	.175	.098
istanbul	.275	.234	.269	.264	.245	.348	.310	.331	.314	.309	.594	.477	.506	.485	.462
kin8fh	.227	.237	.237	.235	.236	.267	.283	.283	.280	.281	.347	.375	.365	.361	.367
kin8fm	.100	.130	.141	.136	.129	.120	.158	.170	.164	.151	.163	.222	.226	.217	.194
kin8nh	.397	.381	.411	.405	.399	.473	.447	.487	.481	.467	.620	.569	.640	.640	.592
kin8nm	.290	.280	.335	.326	.294	.357	.334	.400	.390	.341	.501	.440	.553	.539	.431
laser	.074	.047	.053	.049	.038	.110	.062	.090	.082	.050	.309	.115	.300	.253	.138
mg	.301	.219	.240	.220	.146	.393	.277	.335	.307	.178	.623	.422	.590	.551	.255
mortage	.030	.033	.028	.024	.021	.037	.040	.040	.033	.028	.062	.059	.079	.062	.044
plastic	.496	.526	.516	.510	.536	.573	.613	.597	.584	.600	.715	.788	.755	.735	.711
puma8fh	.457	.466	.465	.465	.444	.553	.548	.557	.560	.535	.737	.725	.738	.740	.745
puma8fm	.207	.265	.209	.208	.201	.253	.313	.252	.251	.242	.346	.413	.343	.344	.326
puma8nh	.447	.455	.432	.432	.413	.549	.542	.529	.530	.507	.756	.765	.716	.719	.696
puma8nm	.210	.339	.185	.180	.175	.259	.405	.226	.221	.215	.367	.586	.315	.306	.301
quakes	.509	.514	.522	.521	.513	.712	.700	.688	.695	.685	1.046	1.169	1.063	1.048	1.058
stock	.122	.104	.100	.091	.085	.149	.122	.130	.118	.103	.206	.159	.217	.188	.145
treasury	.038	.027	.031	.028	.024	.050	.036	.046	.041	.032	.102	.077	.092	.079	.054
wineRed	.452	.402	.428	.422	.426	.560	.521	.531	.528	.529	.821	.738	.772	.739	.726
wineWhite	.396	.368	.372	.367	.366	.493	.462	.459	.454	.456	.706	.671	.707	.698	.670
wizmir	.067	.117	.068	.066	.065	.084	.143	.085	.081	.080	.143	.209	.153	.141	.127
Mean	**.228**	**.241**	**.221**	**.215**	**.206**	**.288**	**.304**	**.279**	**.271**	**.256**	**.442**	**.457**	**.434**	**.413**	**.375**
Mean rank	**3.42**	**3.58**	**3.64**	**2.55**	**1.82**	**3.70**	**3.48**	**3.61**	**2.52**	**1.70**	**3.58**	**3.67**	**3.58**	**2.58**	**1.61**

confidence levels. In addition, RFoa is either significantly or substantially more efficient than the existing methods utilizing separate calibration sets. Again, it is important to note that the two setups utilizing the out-of-bag instances clearly outperformed using a separate calibration set.

Table 7 Normalized nonconformity functions

	90 %	95 %	99 %
RFok versus ANNa	**3E−04**	**2E−06**	**1E−06**
RFok versus kNNc	**4E−10**	**2E−05**	**6E−06**
RFok versus RFia	**3E−14**	**6E−06**	**6E−06**
RFok versus RFoa	0.755	0.156	**0.043**
RFoa versus ANNa	**0.001**	**0.011**	**0.043**
RFoa versus kNNc	**1E−09**	**0.023**	0.072
RFoa versus RFia	**2E−13**	**0.013**	0.072
ANNa versus RFia	**3E−04**	1.678	1.839
ANNa versus kNNc	**0.018**	1.678	1.839
RFia versus kNNc	0.346	1.678	1.839

Adjusted p values. Numbers in bold are significant differences at $\alpha = 0.05$

Summarizing the main experiment, we see that all variants of the novel method produced empirically valid conformal predictors. Most importantly, the suggested approach, i.e., using random forests as the underlying model and utilizing out-of-bag instances for the calibration, clearly outperformed the existing alternatives with regard to efficiency. Finally, when comparing the specific nonconformity functions, the novel, theoretically sound and quite straightforward method to estimate the accuracy of the underlying model based on out-of-bag errors for neighboring instances, actually turned out to be the most efficient.

In order to analyze and explain the results further, the left part of Table 8 shows the accuracy (measured using Root Mean Square Error) for the different underlying models. Looking at mean values and ranks, the most obvious result is that the kNN models are the weakest. We also see that the random forests are generally the most accurate, and that there is a small but systematic advantage in using all the data for the training. The fact that the ANN has a better mean rank, but a worse average error, than RFi is explained by the fact that the two random forest setups have very similar accuracy.

The right part of Table 8 shows the quality of the difficulty estimators. More specifically, the numbers tabulated are the correlations between the estimated difficulty and the actual error made by the underlying model on the test instances. It should be noted that these estimations are calculated in quite different ways by the different setups. ANNa, RFia and RFoa all train a separate model (a linear ANN) to predict the actual error for each instance, while RFok uses the average out-of-bag error from the k nearest neighbors. kNNc, finally, does not explicitly use or model the errors of the underlying model; instead an instance is deemed to be easier if the k nearest neighbors are (relatively) close and agree in their predictions.

Comparing the different estimators, we see that the estimates produced by RFok have the highest correlation with the model errors. The second best is actually kNNc, followed by the approaches using a separate ANN model as estimator.

From this analysis, and the comparison between normalized and standard nonconformity functions above, it is obvious that although the accuracy of the underlying model is very important, the quality of the nonconformity function is vital for the efficiency. Specifically, using normalized nonconformity functions will increase the efficiency and the quality of the difficulty estimations has an apparent impact on the efficiency.

In order to determine the importance of parameter values, a limited post-hoc analysis was performed, see Table 9. As described above, the parameters β, ρ and γ balance the difficulty estimations against the error in the nonconformity functions. In previous studies, all parameter values were set to 0.5, which is not a very robust choice since the importance of the parameter value is heavily affected by the range of the target variable. In this study, as

Table 8 Analysis of underlying models and estimators

Techniques	Underlying model (RMSE)				Estimator (correlation coefficient)				
	ANN	kNN	RFi	RFo	ANNa	kNNc	RFia	RFoa	RFok
abalone	.077	.080	.076	.076	.354	.098	.314	.316	.351
anacalt	.075	.149	.074	.071	.389	.477	.301	.288	.555
bank8fh	.090	.102	.092	.091	.224	.261	.282	.276	.282
bank8fm	.041	.066	.043	.041	.287	.518	.510	.488	.542
bank8nh	.110	.115	.110	.110	.258	.274	.283	.277	.255
bank8nm	.049	.063	.051	.050	.477	.576	.481	.467	.629
boston	.082	.108	.079	.075	.231	.345	.301	.297	.403
comp	.038	.045	.029	.029	.315	.262	.352	.344	.432
concreate	.088	.116	.073	.068	.311	.221	.269	.284	.447
cooling	.072	.081	.049	.049	.546	.638	.687	.698	.829
deltaA	.040	.040	.037	.037	.247	.327	.261	.264	.416
deltaE	.054	.054	.053	.053	.067	.126	.056	.059	.164
friedm	.052	.070	.069	.068	.067	.192	.086	.086	.344
heating	.058	.075	.025	.024	.607	.810	.415	.421	.669
istanbul	.081	.081	.079	.079	.004	.224	−.059	−.088	.108
kin8fh	.070	.075	.075	.074	.215	.193	.249	.246	.210
kin8fm	.031	.042	.046	.044	.156	.246	.312	.309	.483
kin8nh	.121	.119	.126	.125	.114	.195	.128	.129	.247
kin8nm	.090	.088	.105	.101	.149	.199	.158	.157	.466
laser	.038	.049	.036	.033	.334	.595	.394	.381	.567
mg	.110	.099	.089	.083	.379	.677	.344	.342	.706
mortage	.011	.018	.012	.011	.416	.455	.537	.529	.707
plastic	.144	.156	.161	.160	−.008	−.181	.010	.024	−.210
puma8fh	.141	.149	.143	.143	.150	.240	.150	.150	.268
puma8fm	.063	.084	.063	.063	.107	.185	.100	.103	.269
puma8nh	.137	.154	.133	.133	.143	.287	.174	.171	.237
puma8nm	.067	.117	.059	.057	.083	.181	.208	.197	.220
quakes	.173	.175	.174	.174	−.006	.004	.044	.040	.147
stock	.038	.036	.033	.029	.143	.327	.154	.157	.458
treasury	.014	.016	.013	.013	.319	.360	.426	.415	.667
wineRed	.133	.133	.128	.127	.187	.099	.191	.199	.236
wineWhite	.123	.120	.116	.116	.080	.095	.079	.087	.159
wizmir	.024	.046	.022	.021	.214	.345	.144	.165	.181
Mean	**.077**	**.088**	**.075**	**.074**	**.229**	**.298**	**.253**	**.251**	**.377**
Mean Rank	**2.45**	**3.48**	**2.55**	**1.52**	**4.03**	**2.82**	**3.24**	**3.33**	**1.58**

described in Sect. 4.1, the target variables were normalized to [0, 1], so both errors and error estimates take on much smaller values. With this in mind, it was obvious that β too must be smaller. In the experimentation, we set $\beta = 0.01$, based on some initial experimentation. When looking at the efficiencies obtained using different values for β, we see that 0.01 is actually the best choice, even if the difference when compared to $\beta = 0$ is marginal, on

Table 9 The importance of parameter values

Confidence		90 %					95 %					99 %				
ANNa	β	0	.01	.1	.25	.5	0	.01	.1	.25	.5	0	.01	.1	.25	.5
Mean		.228	.228	.236	.240	.242	.288	.288	.299	.304	.307	.444	.442	.456	.465	.470
Mean rank		2.18	1.91	2.82	3.73	4.36	2.42	2.12	2.67	3.48	4.30	2.67	1.85	2.45	3.48	4.55
RFi	β	0	.01	.1	.25	.5	0	.01	.1	.25	.5	0	.01	.1	.25	.5
Mean		.219	.221	.230	.233	.234	.276	.279	.294	.300	.302	.430	.434	.457	.466	.470
Mean Rank		2.15	2.00	2.73	3.61	4.52	2.24	1.82	2.91	3.64	4.39	2.24	2.06	2.64	3.70	4.36
RFoa	β	0	.01	.1	.25	.5	0	.01	.1	.25	.5	0	.01	.1	.25	.5
Mean		.214	.215	.224	.227	.229	.269	.271	.285	.292	.295	.411	.413	.436	.447	.454
Mean rank		1.79	1.73	2.79	3.85	4.85	2.09	1.79	2.67	3.76	4.70	2.12	2.00	2.70	3.64	4.55
RFok	β	0	.01	.1	.25	.5	0	.01	.1	.25	.5	0	.01	.1	.25	.5
Mean		.206	.206	.215	.221	.225	.255	.256	.270	.279	.286	.375	.375	.401	.420	.433
Mean rank		2.00	1.79	2.82	3.73	4.67	2.18	1.94	2.70	3.55	4.64	2.39	2.09	2.42	3.48	4.61
kNNc	$\gamma = \rho$	0	.5	1	1.25	1.5	0	.5	1	1.25	1.5	0	.5	1	1.25	1.5
Mean		.279	.255	.241	.238	.238	.367	.325	.304	.301	.301	.564	.480	.457	.458	.465
Mean rank		4.82	3.82	2.45	1.97	1.94	4.76	3.70	2.33	2.00	2.21	4.61	3.27	2.12	2.18	2.82

Tabulated values are interval sizes, averaged over all data sets

Table 10 Comparison to published results

	Standard				Normalized				
	RFi	RFo	kNN	ANN	RFia	RFoa	RFok	kNNc	ANNa
Confidence 90 %									
abalone	6.63	6.58	6.71	6.60	5.76	5.59	5.61	5.84	5.65
boston	9.67	9.23	13.71	12.44	9.30	8.67	8.12	10.23	11.67
comp	9.26	9.02	10.15	9.75	8.08	7.99	7.91	8.52	8.78
bank	0.08	0.08	0.14	0.06	0.07	0.07	0.07	0.09	0.06
kin	0.51	0.50	0.41		0.50	0.50	0.48	0.40	
puma	4.30	4.20	8.01		4.15	4.13	4.08	7.58	
Confidence 95 %									
abalone	8.95	9.02	9.49	9.08	7.54	7.20	7.30	7.96	7.41
boston	14.45	13.87	19.44	17.18	12.58	11.40	10.52	13.90	16.13
comp	12.00	11.75	13.59	12.34	10.40	10.07	9.94	11.11	10.97
bank	0.11	0.11	0.21	0.08	0.11	0.10	0.11	0.14	0.08
kin	0.61	0.61	0.51		0.60	0.60	0.57	0.49	
puma	5.27	5.14	10.01		5.12	5.03	4.95	9.24	
Confidence 99 %									
abalone	15.00	15.23	16.63	14.97	12.27	12.07	12.28	14.03	12.47
boston	30.17	29.62	38.81	39.32	28.09	23.13	19.00	29.07	32.19
comp	21.44	19.98	22.71	19.86	16.23	15.72	15.60	17.47	17.28
bank	0.20	0.21	0.36	0.14	0.18	0.19	0.19	0.25	0.14
kin	0.84	0.83	0.71		0.84	0.82	0.75	0.67	
puma	7.36	7.35	13.92		7.20	6.94	6.85	12.81	

Interval sizes

most data sets. Larger values for β, on the other hand, clearly reduce the efficiency. So, the conclusion is that although the parameter value β is very important and dependent on the target range, all reasonable values produce conformal predictors with similar efficiency. Looking at the kNNc setup, it should be noted that there are actually two different parameters, making it possible to balance the two different parts of the difficulty estimation. In this study, however, this was not evaluated, instead ρ and γ were both set to 1.0. From the post-hoc analysis, we can see that while larger values (i.e., 1.25 or 1.5) would have increased the efficiency slightly for the lower confidence levels, the resulting differences are almost never large enough to change the ordering of the evaluated setups for specific data sets in the main experiment. In addition, for the confidence level of 99 %, $\rho = \gamma = 1.0$ was actually the best parameter setting.

In order to directly compare the efficiency of our methods to the published results in Papadopoulos and Haralambous (2011) and Papadopoulos et al. (2011), Table 10 compares the interval sizes obtained in Experiment 2 to the interval sizes published. Starting with the standard nonconformity functions, we immediately see that RFi and RFo almost always produced smaller prediction intervals than kNN. For several data sets and confidence levels, the differences are quite large. From a direct comparison, it is quite obvious that both RFo and RFi were more efficient than kNN, winning five of six data sets on all confidence levels. When compared to ANN, however, the results vary over the different confidence levels.

For normalized nonconformity functions, we see that RFoa and, in particular, RFok are the most efficient over all data sets and confidence levels. Counting wins and losses, RFoa and RFok outperformed both kNNc and ANNa on a majority of data sets. To summarize this comparison with published results, we see that the results from the main experiment are confirmed, i.e., conformal regressors based on random forests, especially when utilizing out-of-bag instances for the calibration, outperform the existing techniques, both when using standard and normalized nonconformity functions.

5 Concluding remarks

In this paper, the use of random forests has been proposed as a strong candidate for regression conformal prediction, since it allows for the necessary calibration to be performed on the out-of-bag examples, thus making it possible to utilize all available data as a proper training set. In one of the largest empirical evaluations to date on regression conformal prediction, the random forest approach was compared to existing state-of-the-art approaches, based on ANN and kNN, for both the standard and normalized settings, i.e., when generating prediction intervals of uniform and varying sizes, respectively. The results show that the suggested approach, on almost all confidence levels and using both standard and normalized nonconformity functions, produced significantly more efficient conformal predictors than the existing alternatives. In particular, the most efficient setup overall was found to be one suggested in this paper, i.e., a random forest conformal predictor calibrated using a normalized nonconformity function based on out-of-bag errors of neighboring instances. The empirical evidence hence strongly suggests that random forests in conjunction with out-of-bag calibration is a highly competitive conformal regressor.

There are several possible directions for future research. One direction concerns the type of model to use for estimating the difficulty of each instance in the normalized setting. Currently, fairly simple models have been evaluated, i.e., kNN and linear ANNs, and gains are to be expected from considering more elaborate techniques, as well as from performing further parameter tuning. Another direction concerns investigating the application of other state-of-the-art machine learning algorithms, e.g., SVMs, to the regression conformal prediction framework and comparing the resulting conformal predictors to random forests with out-of-bag calibration.

Acknowledgments We thank the anonymous reviewers for many helpful comments and suggestions. This work was supported by the Swedish Foundation for Strategic Research through the project High-Performance Data Mining for Drug Effect Detection (IIS11-0053) and the Knowledge Foundation through the project Big Data Analytics by Online Ensemble Learning (20120192).

References

Alcalá-Fdez, J., Fernández, A., Luengo, J., Derrac, J., & García, S. (2011). Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *Multiple-Valued Logic and Soft Computing*, *17*(2–3), 255–287.

Bache, K., & Lichman, M. (2013). UCI machine learning repository, URL http://archive.ics.uci.edu/ml.

Bergmann, B., & Hommel, G. (1988). Improvements of general multiple test procedures for redundant systems of hypotheses. In *Multiple hypotheses testing* (pp. 100–115). New York: Springer.

Bhattacharyya, S. (2011). Confidence in predictions from random tree ensembles. In: *IEEE ICDM* (pp. 71–80).

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and regression trees*. Belmont: Wadsworth.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*(2), 123–140.
Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32.
Caruana, R., & Niculescu-Mizil, A. (2006). An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd International Conference on Machine Learning* (pp. 161–168). ACM.
Devetyarov, D., & Nouretdinov, I. (2010). Prediction with confidence based on a random forest classifier. *Artificial Intelligence Applications and Innovations*, *7*, 37–44.
Devetyarov, D., Nouretdinov, I., Burford, B., Camuzeaux, S., Gentry-Maharaj, A., Tiss, A., et al. (2012). Conformal predictors in early diagnostics of ovarian and breast cancers. *Progress in Artificial Intelligence*, *1*(3), 245–257.
Flake, G. W., & Lawrence, S. (2002). Efficient svm regression training with smo. *Machine Learning*, *46*(1–3), 271–290.
Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of American Statistical Association*, *32*, 675–701.
Gammerman, A., Vovk, V., & Vapnik, V. (1998). Learning by transduction. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence* (pp. 148–155). Burlington: Morgan Kaufmann.
Garcıa, S., & Herrera, F. (2008). An extension on statistical comparisons of classifiers over multiple data sets for all pairwise comparisons. *Journal of Machine Learning Research*, *9*(2677–2694), 66.
Ho, T. K. (1998). The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *20*(8), 832–844.
Johansson, U., Boström, H., & Löfström, T. (2013a). Conformal prediction using decision trees. In *IEEE International Conference on Data Mining* (pp. 330–339).
Johansson, U., König, R., Löfström, T., & Boström, H. (2013b). Evolved decision trees as conformal predictors. In *IEEE Congress on Evolutionary Computation* (pp. 1794–1801).
Lambrou, A., Papadopoulos, H., & Gammerman, A. (2011). Reliable confidence measures for medical diagnosis with evolutionary algorithms. *IEEE Transactions on Information Technology in Biomedicine*, *15*(1), 93–99.
Löfström, T., Johansson, U., & Boström, H. (2013). Effective utilization of data in inductive conformal prediction. In *The IEEE 2013 International Joint Conference on Neural Networks (IJCNN)*.
Makili, L., Vega, J., Dormido-Canto, S., Pastor, I., & Murari, A. (2011). Computationally efficient svm multi-class image recognition with confidence measures. *Fusion Engineering and Design*, *86*(6), 1213–1216.
Nguyen, K., & Z, Luo. (2012). Conformal prediction for indoor localisation with fingerprinting method. *Artificial Intelligence Applications and Innovations*, *3*, 214–223.
Nouretdinov, I., Vovk, V., Vyugin, M., & Gammerman, A. (2001). Pattern recognition and density estimation under the general i.i.d. assumption. *Computational Learning Theory* (pp. 337–353)., volume 2111 of Lecture Notes in Computer Science Berlin Heidelberg: Springer.
Papadopoulos, H. (2008). Inductive conformal prediction: Theory and application to neural networks. *Tools in Artificial Intelligence*, *18*(315–330), 2.
Papadopoulos, H., & Haralambous, H. (2010). Neural networks regression inductive conformal predictor and its application to total electron content prediction. *Artificial Neural Networks ICANN 2010* (pp. 32–41)., volume 6352 of Lecture Notes in Computer Science Berlin Heidelberg: Springer.
Papadopoulos, H., & Haralambous, H. (2011). Reliable prediction intervals with regression neural networks. *Neural Networks*, *24*(8), 842–851.
Papadopoulos, H., Proedrou, K., Vovk, V., & Gammerman, A. (2002). Inductive confidence machines for regression. In *Machine Learning: ECML* 2002 (pp. 345–356). New York: Springer.
Papadopoulos, H., Gammerman, A., & Vovk, V. (2009a). Reliable diagnosis of acute abdominal pain with conformal prediction. *Engineering Intelligent Systems*, *17*(2), 127.
Papadopoulos, H., Papatheocharous, E., & Andreou, A. S. (2009b). Reliable confidence intervals for software effort estimation. In *AIAI Workshops* (pp. 211–220).
Papadopoulos, H., Vovk, V., & Gammerman, A. (2011). Regression conformal prediction with nearest neighbours. *Journal of Artificial Intelligence Research*, *40*(1), 815–840.
Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, *1*, 81–106.
Rasmussen, C. E., & Christopher, K. I. (2005). *Gaussian processes for machine learning*. Cambridge: MIT Press.
Rasmussen, C. E., Neal, R. M., Hinton, G. E., van Camp, D., Revow, M., Ghahramani, Z., Kustra, R., & Tibshirani, R. (1996). Delve data for evaluating learning in valid experiments. www.cs.toronto.edu/delve.
Valiant, L. G. (1984). A theory of the learnable. *Communications of the ACM*, *27*(11), 1134–1142.
Vovk, V., Gammerman, A., & Shafer, G. (2006). *Algorithmic learning in a random world*. New York: Springer.