Evaluation of a Variance-Based Nonconformity Measure for Regression Forests

Henrik Boström[1(✉)], Henrik Linusson[2], Tuve Löfström[2], and Ulf Johansson[3]

[1] Department of Computer and Systems Sciences, Stockholm University, Kista, Sweden
henrik.bostrom@dsv.su.se

[2] Department of Information Technology, University of Borås, Borås, Sweden
{henrik.linusson,tuve.lofstrom}@hb.se

[3] Department of Computer Science and Informatics, Jönköping University, Jönköping, Sweden
ulf.johansson@ju.se

Abstract. In a previous large-scale empirical evaluation of conformal regression approaches, random forests using out-of-bag instances for calibration together with a k-nearest neighbor-based nonconformity measure, was shown to obtain state-of-the-art performance with respect to efficiency, i.e., average size of prediction regions. However, the use of the nearest-neighbor procedure not only requires that all training data have to be retained in conjunction with the underlying model, but also that a significant computational overhead is incurred, during both training and testing. In this study, a more straightforward nonconformity measure is investigated, where the difficulty estimate employed for normalization is based on the variance of the predictions made by the trees in a forest. A large-scale empirical evaluation is presented, showing that both the nearest-neighbor-based and the variance-based measures significantly outperform a standard (non-normalized) nonconformity measure, while no significant difference in efficiency between the two normalized approaches is observed. Moreover, the evaluation shows that state-of-the-art performance is achieved by the variance-based measure at a computational cost that is several orders of magnitude lower than when employing the nearest-neighbor-based nonconformity measure.

Keywords: Conformal prediction · Nonconformity measures · Regression · Random forests

1 Introduction

When employing the conformal prediction (CP) framework [13], the probability of making incorrect predictions is bounded by a user-provided confidence threshold. Rather than just providing a single bound on the prediction error for examples drawn from the underlying distribution, CP allows for providing different bounds for different instances, something which may be valuable in many different scenarios. For example, in the medical domain, the ability to assess the confidence in predictions related to individual patients, rather than at the group level, may be crucial input for decisions concerning alternative treatments for a specific patient.

CP relies on real-valued functions, called nonconformity measures, that provide estimates for how different a new example is from a set of old examples. It is possible to design many different nonconformity functions for a specific predictive model, and each will result in a different conformal predictor. All such conformal predictors will be *valid* i.e., the probability of excluding the correct label will be less than one minus the confidence level. However, there may be significant differences in terms of *efficiency*, i.e., the sizes of output prediction regions, meaning that the informativeness of the output of different conformal predictors may vary substantially. For classification, efficiency is often measured as the (average) number of labels present in the prediction sets, while for regression, which is the focus of this paper, efficiency is most commonly measured as the (average) size of the intervals.

CP was originally introduced as a transductive approach [5], which requires the learning of a new model for each new test instance to be predicted. Since this in many cases may be computationally prohibitive, inductive conformal prediction (ICP) was suggested [13]. In ICP, which is the focus of this study, a single model is learned from the training data and that model is then used for predicting all test instances. In ICP, however, the calculation of the nonconformity scores (normally) requires comparing predicted values with true target values not seen during training, and the standard procedure to achieve this is to set aside a separate subset of the training examples, called the *calibration set*. However, when the underlying model is an ensemble constructed using bagging, such as a random forest [3], there is also an option to use out-of-bag estimates for the calibration, effectively allowing all training data to be used for constructing the underlying model, something which has been exploited in the context of ICP for bagged ANNs [7] and random forests [6].

Until recently, most studies on ICP conformal regression have focused on one specific underlying model, using a limited number of data sets, making them serve mainly as proofs-of-concept rather than allowing for drawing statistically valid conclusions; see e.g., [8,10]. The apparent need for larger studies evaluating techniques for producing efficient conformal predictors, motivated the study in [6], in which the use of a random forest as the underlying model was compared to existing state-of-the-art conformal regressors, based on neural networks [9] and k-nearest neighbors [11]. A number of nonconformity measures were investigated, including the option to use out-of-bag estimates for the necessary calibration. The results in [6] showed that for almost all confidence levels and using both standard and normalized nonconformity functions, a random forest conformal predictor calibrated using a normalized nonconformity function based on out-of-bag errors of neighboring instances, produced significantly more efficient conformal predictors than the existing alternatives.

However, the use of a nonconformity measure based on the k nearest neighbors requires that access has to be provided to all training instances even at the time the model is deployed, something which occasionally may limit the usefulness of the approach, e.g., when there are size constraints, such as on mobile devices, or when data is highly sensitive and may not be re-distributed. An even more important constraint may be the computation time, both for training and testing. The computational cost of calculating the average error of the k nearest neighbors for each example in the training set is quadratic in the number of examples, hence incurring a substantial additional cost for employing the conformal framework, which may be a limiting factor in particular when handling large training sets. Even for testing, there is an additional cost when using the nearest neighbor nonconformity measure, since the distance of each test instance to all training instances needs to be calculated. To increase the applicability of conformal regression using random forests, there is hence a need for nonconformity measures with lower computational cost. One such candidate approach is to estimate the difficulty of an instance, not by averaging the errors of its neighbors, but by utilizing the fact that each prediction of a random forest is formed by averaging votes of the individual trees in the forest. For difficult cases, one would expect a larger degree of disagreement among the trees, i.e., a higher variance among the individual predictions, than for easier cases. In other words, the variance could be used as an estimate of the difficulty. In fact, this idea is not entirely novel, but was already investigated for k-nearest neighbor regressors in [11], where the variance of the target value of the k neighbors was one of several proposed estimates of difficulty. The main question of this study is whether or not this is an effective approach for forests of regression trees.

In the next section, we formalize the conformal regression framework. In Sect. 3, we describe the current state-of-the-art approach for conformal regression, i.e., random forests using out-of-bag errors of neighboring instances, as well as the proposed approach, which instead of employing the nearest neighbor procedure uses the variance of the predictions made by the individual trees to normalize the prediction regions. 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

Predictions of a conformal regressor take the form of real-valued intervals (a, b), where $P(a \leq y \leq b) \geq 1 - \delta$ for a test pattern $\boldsymbol{x}$ with true output value y and a user-specified significance level δ. To produce such *prediction intervals*, a conformal regressor utilizes a *nonconformity measure*, which is a real-valued function that measures the strangeness of an example $(\boldsymbol{x}, y)$. This nonconformity measure is typically based on the prediction error of a traditional machine learning model, called the *underlying model* of the conformal regressor. Based on the nonconformity scores of examples with known labels, a conformal predictor uses hypothesis testing to reject (or fail to reject) tentative output values $\tilde{y} \in \mathbb{R}$ at significance δ. For regression problems, the nonconformity measure is most often simply the absolute prediction error [9–11],

$$\alpha_i = A(\boldsymbol{x}_i, y_i, h) = |y_i - \hat{y}_i| = |y_i - h(\boldsymbol{x}_i)|, \tag{1}$$

where h is the underlying model trained on the problem in question, e.g., a regression tree, a neural network or an ensemble model.

To train an inductive conformal regressor, the following procedure is normally used:

1. Divide the training set $Z = \{(\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_l, y_l)\}$ into two disjoint subsets Z^t (a proper training set) and Z^c (a calibration set):
   - $Z^t = \{(\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_m, y_m)\}$
   - $Z^c = \{(\boldsymbol{x}_{m+1}, y_{m+1}), ..., (\boldsymbol{x}_l, y_l)\}$
2. Train the underlying model h using Z^t.
3. Use the nonconformity measure, e.g. (1), to measure the nonconformity of the examples in Z^c, obtaining a list, sorted in descending order, of calibration scores $S = \alpha_1, ..., \alpha_q$ where $q = |Z^c|$.

When a new test instance $\boldsymbol{x}_j$ arrives, a prediction region is constructed as follows:

1. Obtain a prediction $\hat{y}_j = h(\boldsymbol{x}_j)$.
2. Find the calibration score $\alpha_{s(\delta)}$ where $s(\delta) = \lfloor \delta(q+1) \rfloor$.
3. Using the (partial) inverse of the nonconformity measure, obtain the largest error that is consistent with δ, i.e., $A^{-1}(\alpha_{s(\delta)})$. This is the maximum error made by h on $\boldsymbol{x}_j$ with confidence $1 - \delta$.

If the nonconformity measure in (1) is used, the predictive step simply translates into a prediction region for $\boldsymbol{x}_j$ being constructed as

$$\hat{Y}_j^\delta = \hat{y}_j \pm \alpha_{s(\delta)}, \tag{2}$$

since, with probability $1 - \delta$, the underlying model h will not make an absolute prediction error greater than $\alpha_{s(\delta)}$.

It must be noted that when using (1) and (2), the conformal regressor will, for any specific significance level δ, always produce prediction intervals of the same size for every $\boldsymbol{x}_j$; i.e., the error bounds will not be dependent on properties of a specific test instance. It is, however, possible to introduce *normalized* nonconformity measures, where the absolute error is divided by a term σ_i that is dependent on the prediction instance, usually corresponding to the estimated difficulty of the underlying model for making a correct prediction for that instance; see e.g., [9,11]:

$$\alpha_i = \frac{|y_i - \hat{y}_i|}{\sigma_i}. \tag{3}$$

With normalized nonconformity measures, the prediction interval for $\boldsymbol{x}_j$ is:

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

The motivation for employing normalized nonconformity functions is that instances estimated to be easier to predict will be assigned narrower intervals than instances that are judged to be more difficult. It should be noted that there are several ways to estimate the difficulty; one suggestion is to train another model for predicting the errors; see e.g., [9]. Other approaches use properties of the underlying model; see e.g., [11].

3 Methods

In this section, we describe the approach for regression conformal prediction using random forests. In particular, we describe three nonconformity measures that will be compared in the empirical investigation: (i) a standard (non-normalized) nonconformity measure, (ii) a nonconformity measure where the difficulty is estimated by the average error of the nearest neighbors, which was shown to result in state-of-the-art performance in [6], and (iii) a variance-based nonconformity measure, originally proposed for k-nearest neighbor classifiers in [11], which previously has not been evaluated for random forests.

3.1 Regression Conformal Prediction using Random Forests

A random forest [3] is an ensemble consisting of *random trees*, which are decision trees generated in a specific way. In order to introduce the necessary diversity, each random tree is trained on a *bootstrap replicate* [2], and only a randomized subset of the attributes are available for the algorithm when optimizing each interior split. The instances that were missing in the bootstrap replicate, for a specific tree, are said to be *out-of-bag* (oob) for that tree. In this study, and similar to [6], we will investigate nonconformity functions that are based on absolute errors, see (1) and (3), where oob instances are used for calculating calibration scores, instead of using a separate calibration set. This means that for each instance in the original training set, only those trees in the generated forest for which the instance is oob, are used for generating the prediction, i.e., instead of $\hat{y}_i = h(\boldsymbol{x}_i)$ in (1) and (3), where h is a random forest, $\hat{y}_i = h_i(\boldsymbol{x}_i)$, where $h_i \subseteq h$. The expected number of trees used to form an oob prediction is approximately 0.368 of the original number of trees, since the probability of including a training example in a bootstrap replicate is about 0.632 [2], leading to that prediction errors on the oob instances can be expected to be at least as large as for independent test instances when using the entire forest, since the underlying model used for the calibration is weaker. Hence, as argued in [6], calculating non-conformity scores using oob instances should lead to valid, although conservative, prediction regions. It should be noted that since all training data can be used for constructing the underlying models, these are typically stronger than the corresponding models trained on a subset, i.e., when excluding the calibration instances, something which was demonstrated in [6] to result in significant efficiency improvements.

3.2 Non-normalized Nonconformity Measure

The first nonconformity measure employs (2), i.e., there is no normalization, so all prediction regions will have identical sizes. It must be noted, however, that out-of-bag instances are used for the calibration instead of a separate calibration set, making it possible to use all available instances for both the training and the calibration. More specifically, when producing the nonconformity score for a calibration instance $\boldsymbol{z}_i$, the ensemble used for producing the prediction $\hat{y}_i$ consists of all trees that were not trained using $\boldsymbol{z}_i$, i.e., $\boldsymbol{z}_i$ was out-of-bag for those trees.

3.3 Nearest Neighbor-Based Normalization

The second nonconformity measure employs normalization using (3), i.e., the sizes of the prediction regions vary depending on the estimated difficulty of the instances. Inspired by nonconformity measures proposed for k-nearest neighbor classifiers [11], this nonconformity measure estimates difficulty by the (out-of-bag) error of the k nearest neighbors, with the obvious motivation that low errors for neighboring instances imply a relatively easy part of the feature space. The exact number of neighbors to use is optimized (between 1 and 45) for each training set (more precisely, for each fold, when performing cross-validation), and the k resulting in the smallest average interval size of the resulting conformal regressor is chosen.

The resulting nonconformity measure for an instance $(\boldsymbol{x}_i, y_i)$ is:

$$\alpha_i = \frac{|y_i - \hat{y}_i|}{\mu_i + \beta} \tag{5}$$

where μ_i is an estimate of the difficulty and β is a parameter, used to control the sensitivity of the nonconformity measure. The difficulty estimate for this particular nonconformity measure is the average, distance-weighted, out-of-bag error of the k nearest neighbors:

$$\mu_i = \frac{\sum_{n=1}^{k} o_n/d_n}{\sum_{n=1}^{k} 1/d_n} \tag{6}$$

where $\{o_1, \ldots, o_k\}$ are the out-of-bag errors of the k nearest neighbors and $\{d_1, \ldots, d_k\}$ are the Euclidean distances of the nearest neighbors to $\boldsymbol{x}_i$ plus a small term ϵ (to avoid division by zero).

Using this nonconformity function, the prediction intervals become:

$$\hat{Y}_j^\delta = \hat{y}_j \pm \alpha_{s(\delta)}(\mu_j + \beta) \tag{7}$$

When used with random forests and out-of-bag calibration, this nonconformity measure was in [6] shown to outperform all competing approaches, including conformal regressors based on neural networks [9] and k-nearest neighbors [11]. Hence, this particular configuration may be considered as the current state-of-the-art for inductive conformal regression.

3.4 Variance-Based Normalization

The third, and last, nonconformity measure that is evaluated in this study estimates difficulty by the variance of the predictions of the individual trees in the forest. The motivation for this difficulty estimator is that for easier instances, one may expect a higher degree of agreement among the trees in the forest. This nonconformity measure has, again, been studied in the context of k-nearest neighbor classifiers [11], but has not previously been investigated for conformal regressors using random forests. This measure is on the same form as the previous (5), but where μ_i now corresponds to the variance of the individual predictions for an instance $(\boldsymbol{x}_i, y_i)$:

$$\mu_i = \frac{\sum_{n=1}^{s} p_n^2}{s} - \left(\frac{\sum_{n=1}^{s} p_n}{s}\right)^2 \tag{8}$$

where $\{p_1, \ldots, p_s\}$ are the predictions of the trees in the forest for which the instance $(\boldsymbol{x}_i, y_i)$ is out-of-bag.

Using this nonconformity measure, the prediction intervals are, as for the previous measure, calculated using (7).

4 Empirical Evaluation

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

4.1 Experimental Setup

For the empirical investigation, all competing methods were re-implemented in the Julia language[1], and a large-scale study, using 33 publicly available data sets from the UCI [1] and Delve [12] repositories, was performed. The considered data sets are small to medium sized; ranging from approximately 500 to 10000 instances. To allow for comparing sizes of prediction regions with the entire output space, the target variable was normalized for each dataset by:

$$\tilde{y}_i = \frac{y_{max} - y_i}{y_{max} - y_{min}} \tag{9}$$

where y_{max} and y_{min} are the highest and lowest output values, respectively, for the dataset. The same normalization was employed also for each input variable, to avoid choice of scale having an impact when calculating Euclidean distances for the nearest neighbor-based nonconformity measure. The latter has neither any effect on the other nonconformity measures nor on the underlying random forest models, i.e., the predictive performance is unaffected.

Regarding parameter values, similar settings as in [6] were employed for all data sets and methods. Specifically, all random forests consisted of 500 random trees, the sensitivity parameter β was set to 0.01 while the parameter ϵ was set to 0.001. A ten-fold cross validation scheme was adopted with all reported values being averaged over the ten folds. Results are reported for three confidence levels: 90 %, 95 % and 99 %.

For each method and dataset 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, are measured. For valid conformal predictors, the error rate should not (in the long run) be higher than one minus the chosen confidence threshold. Hence, by investigating the error rate, 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

[1] www.julialang.org.

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 similar to [6] opted for comparing the average sizes over all prediction regions.

In order to allow for a comparison of the computational cost for generating and applying the different nonconformity measures, i.e., during training and testing, respectively, the CPU times for these activities were recorded, separately from the time taken to build the forests and obtaining predictions from the individual trees. In the experiment, a DELL T7910 with two 14-core 2.6 GHz CPUs (E5-2697v3) with 64 GB RAM was employed. The generation and application of all nonconformity measures was performed on a single core only, while the forest construction and predictions utilized all cores in parallel[2].

To analyze any differences in efficiency between the two normalized approaches, the correlation coefficient between the estimated difficulty of the test instances and the actual prediction error are reported for each method. The expectation is that a higher correlation leads to more efficient predictions.

4.2 Experimental Results

Table 1 shows the error rates, i.e., the fraction of test instances for which the true target value falls outside the predicted region, of three methods utilizing different nonconformity functions: using no normalization (M1); using nearest-neighbor normalization (M2); and, using variance-based normalization (M3). Looking at these results, it is apparent that all three methods behave as expected for valid predictors: the error rates, for each data set, lie close to the predetermined significance level.

A statistical analysis of the error rates at the three confidence levels presented (90 %, 95 % and 99 %), using a Friedman test followed by a Nemenyi post-hoc test (with alpha=0.05) [4], shows that: (i) M3 has a significantly lower error rate than both M1 and M2 for the 90 % level, (ii) M3 has a significantly lower error rate than M1 for the 95 % level, and (iii) M3 has a significantly lower error rate than M2 for the 99 % confidence level. Hence, the variance-based approach clearly seems to be the most conservative of the three methods.

Looking at the interval sizes tabulated in Table 2, 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 best method at the 90 % confidence level returned prediction regions covering, approximately, 21 % of the range. The corresponding average values for the 95 % and 99 % confidence levels are (approximately) 26 % and 38 %, respectively. Clearly, these prediction regions must be considered informative. An analysis of the interval sizes, using the same statistical test as earlier, reveals that there is no significant difference between M2 and M3 for any of the three confidence levels, while both

[2] The Julia implementation can be obtained from the first author upon request.

Table 1. Error rates

Confidence dataset \ Technique	0.90			0.95			0.99		
	M1	M2	M3	M1	M2	M3	M1	M2	M3
abalone	.099	.101	.104	.050	.053	.049	.010	.013	.010
anacalt	.099	.082	.094	.047	.036	.047	.008	.012	.009
bank8fh	.100	.099	.098	.049	.050	.047	.011	.011	.009
bank8fm	.099	.098	.093	.049	.049	.048	.010	.010	.009
bank8nh	.100	.101	.098	.050	.051	.050	.010	.011	.010
bank8nm	.100	.102	.098	.050	.051	.048	.009	.011	.010
boston	.107	.101	.099	.049	.042	.036	.008	.010	.010
comp	.096	.100	.098	.049	.050	.050	.010	.011	.010
concreate	.098	.081	.100	.050	.044	.049	.010	.008	.008
cooling	.095	.092	.092	.052	.050	.050	.012	.013	.012
deltaA	.101	.103	.100	.050	.051	.049	.009	.010	.010
deltaE	.099	.103	.099	.051	.053	.048	.010	.012	.010
friedm	.097	.098	.093	.050	.046	.050	.008	.004	.007
heating	.102	.081	.092	.050	.048	.053	.005	.006	.009
istanbul	.105	.108	.099	.050	.052	.050	.007	.011	.007
kin8fh	.099	.098	.099	.050	.049	.049	.010	.009	.009
kin8fm	.099	.094	.094	.049	.043	.047	.010	.007	.009
kin8nh	.099	.100	.098	.049	.048	.048	.009	.009	.008
kin8nm	.096	.092	.096	.049	.047	.047	.010	.009	.008
laser	.098	.088	.090	.047	.041	.049	.009	.009	.007
mg	.097	.097	.095	.046	.055	.051	.009	.013	.012
mortage	.091	.087	.091	.044	.034	.044	.009	.007	.008
plastic	.101	.107	.098	.052	.050	.050	.008	.015	.007
puma8fh	.097	.100	.097	.050	.051	.048	.009	.011	.010
puma8fm	.100	.099	.100	.050	.051	.049	.009	.010	.008
puma8nh	.100	.102	.096	.051	.050	.047	.010	.010	.009
puma8nm	.095	.096	.095	.048	.049	.046	.009	.011	.009
quakes	.100	.107	.096	.051	.060	.053	.014	.026	.019
stock	.094	.088	.099	.046	.040	.046	.008	.003	.009
treasury	.099	.095	.103	.048	.042	.045	.011	.012	.010
wineRed	.101	.104	.098	.051	.054	.048	.009	.014	.010
wineWhite	.103	.107	.101	.048	.053	.047	.011	.011	.008
wizmir	.095	.106	.089	.047	.046	.045	.010	.012	.012
Mean	.099	.097	.097	.049	.048	.048	.009	.011	.009
Mean rank	2.26	2.21	1.53	2.32	2.03	1.65	1.95	2.39	1.65

Table 2. Region sizes.

Confidence dataset \ Technique	0.90			0.95			0.99		
	M1	M2	M3	M1	M2	M3	M1	M2	M3
abalone	.234	.214	.214	.321	.274	.282	.544	.463	.495
anacalt	.139	.081	.107	.258	.092	.126	.501	.190	.221
bank8fh	.300	.290	.268	.377	.361	.342	.533	.538	.585
bank8fm	.139	.131	.123	.175	.158	.145	.251	.211	.191
bank8nh	.322	.307	.281	.447	.420	.414	.789	.744	.782
bank8nm	.145	.121	.111	.210	.160	.141	.399	.245	.217
boston	.193	.192	.200	.276	.254	.253	.605	.432	.418
comp	.086	.077	.083	.114	.098	.107	.187	.153	.170
concreate	.204	.208	.184	.258	.270	.235	.475	.473	.362
cooling	.170	.107	.150	.216	.124	.184	.287	.146	.243
deltaA	.117	.108	.113	.154	.139	.141	.260	.212	.228
deltaE	.174	.170	.172	.215	.215	.214	.315	.305	.304
friedm	.215	.205	.217	.258	.243	.269	.360	.319	.406
heating	.070	.058	.065	.087	.068	.078	.168	.094	.102
istanbul	.260	.247	.257	.318	.315	.336	.491	.497	.494
kin8fh	.241	.240	.240	.291	.285	.285	.398	.372	.375
kin8fm	.134	.123	.132	.166	.144	.160	.245	.183	.218
kin8nh	.413	.404	.408	.488	.472	.478	.622	.595	.613
kin8nm	.331	.303	.321	.396	.350	.374	.527	.445	.478
laser	.044	.039	.041	.085	.054	.059	.330	.150	.141
mg	.243	.172	.163	.341	.221	.201	.596	.322	.336
mortage	.022	.019	.021	.036	.027	.032	.073	.044	.059
plastic	.549	.545	.592	.644	.637	.734	.807	.851	.943
puma8fh	.470	.446	.444	.565	.532	.529	.741	.724	.754
puma8fm	.210	.204	.201	.254	.243	.240	.341	.323	.322
puma8nh	.438	.427	.416	.543	.518	.503	.731	.697	.697
puma8nm	.202	.199	.201	.243	.238	.233	.345	.328	.310
quakes	.556	.540	.605	.705	.681	.751	1.000	.900	.942
stock	.076	.074	.074	.093	.089	.088	.158	.131	.124
treasury	.026	.022	.025	.042	.030	.039	.088	.051	.071
wineRed	.366	.375	.336	.495	.499	.452	.734	.721	.636
wineWhite	.321	.320	.289	.416	.420	.372	.644	.662	.551
wizmir	.059	.058	.059	.074	.072	.073	.139	.126	.125
Mean	.226	.213	.216	.290	.264	.269	.445	.383	.391
Mean rank	2.79	1.42	1.79	2.76	1.61	1.64	2.73	1.55	1.73

M2 and M3 result in significantly smaller interval sizes than M1 for all three confidence levels (with p-values much smaller than 0.01).

Table 3 displays execution times for the three different methods tested. First listed is the total time (in seconds) for the tasks common to all methods of training the underlying model (random forest), collecting out-of-bag predictions and obtaining the individual predictions for the test instances. As expected, only small variations are observed, since these tasks are identical for all three approaches.

Second, the total number of seconds required to generate the nonconformity measure using the out-of-bag instances is listed. Here, there is a clear difference between the three methods. M1 requires only that the errors on the out-of-bag instances are computed and ordered, which is a fairly quick operation. M2, on the other hand, requires an extra (particularly costly) step of making, for each out-of-bag instance, an additional prediction using the nearest-neighbor procedure to calculate the normalization term of the nonconformity measure. Finally, M3, for which normalization does not require any additional predictive step, the calculation of nonconformity scores comes with very little overhead compared to the non-normalized variant M1.

Listed in the third column is the total time (in seconds) required to calculate prediction regions for the test set. Again, the time required for making predictions using the variance-based M3 is only marginally longer than for the non-normalized M1, while M2 again incurs a very large overhead.

It should be noted that the observed execution times are dependent on the particular implementation of the algorithms, and possibly some of the performance differences could be reduced by carefully optimizing the code. However, there is an inherent difference in computational complexity of the underlying algorithms, which will not disappear even with smarter implementations. Comparing the computational cost that is specific to performing conformal prediction, i.e., not including the time for building and obtaining predictions from the underlying model, the variance-based approach is in this experiment several orders of magnitude faster than the nearest neighbor approach (the former is on average over twenty thousand times faster than the latter) and this gap will most likely remain wide even with a substantially more efficient implementation of the k-nearest neighbor procedure.

Finally, in order to investigate how well the difficulty estimates employed by the nearest-neighbor and the variance-based approaches actually work, we investigated the correlation coefficients between $\mu_i + \beta$ and the test error for the two normalized approaches. The results are displayed in Table 4. When testing for significant differences, the p-value is 0.056 in favor of M3 over M2, hence indicating that variance in fact may be a more effective way of ordering instances according to expected test error than employing the nearest-neighbor procedure. This difference obviously does not directly carry over to a corresponding difference in region size, as the latter was found above to be insignificant (Table 2). However, the importance of correctly ranking the instances according to difficulty is demonstrated by the fact that the method with the highest correlation

Table 3. Time taken (in seconds) to build and obtain predictions from the underlying models (identical tasks for all methods), to generate the nonconformity functions and to calculate prediction regions for the test set

Dataset \ Technique	Common tasks			Calibration			Application		
	M1	M2	M3	M1	M2	M3	M1	M2	M3
abalone	2.91	3.02	2.98	.002	35.1	.003	.000	4.00	.000
anacalt	1.02	.98	1.03	.002	31.8	.003	.000	3.62	.000
bank8fh	6.54	6.60	6.74	.003	151.8	.005	.000	17.19	.000
bank8fm	6.67	6.67	6.78	.005	150.3	.005	.000	17.11	.000
bank8nh	6.47	6.54	6.49	.005	151.0	.006	.000	17.24	.000
bank8nm	6.36	6.43	6.53	.003	150.9	.005	.000	17.21	.000
boston	.34	.33	.33	.000	.4	.000	.000	.05	.000
comp	6.67	6.83	6.90	.003	150.3	.007	.000	17.17	.000
concreate	.63	.64	.62	.000	1.7	.001	.000	.19	.000
cooling	.26	.27	.27	.000	1.0	.001	.000	.11	.000
deltaA	5.35	5.38	5.31	.011	108.5	.005	.000	12.40	.000
deltaE	7.15	7.36	7.37	.004	207.1	.007	.000	23.78	.001
friedm	.77	.78	.77	.000	2.3	.001	.000	.27	.000
heating	.27	.28	.27	.000	.9	.001	.000	.10	.000
istanbul	.36	.36	.36	.000	.5	.000	.000	.05	.000
kin8fh	5.93	5.96	6.19	.003	148.0	.007	.000	16.91	.000
kin8fm	5.94	6.06	6.09	.003	147.6	.005	.000	16.85	.001
kin8nh	6.21	6.26	6.29	.003	147.3	.006	.000	16.89	.000
kin8nm	6.07	6.10	6.18	.003	149.2	.007	.000	17.07	.000
laser	.64	.63	.63	.000	1.7	.001	.000	.19	.000
mg	.93	.94	.95	.001	3.2	.001	.000	.36	.000
mortage	.74	.73	.74	.000	1.8	.001	.000	.20	.000
plastic	.49	.50	.49	.001	4.7	.001	.000	.53	.000
puma8fh	6.24	6.31	6.33	.004	149.6	.006	.000	17.04	.000
puma8fm	6.51	6.26	6.26	.003	150.0	.005	.000	17.14	.000
puma8nh	6.22	6.33	6.21	.005	148.7	.006	.000	17.00	.000
puma8nm	6.11	6.22	6.19	.003	146.2	.005	.000	16.71	.000
quakes	1.53	1.48	1.49	.001	8.3	.001	.000	.94	.000
stock	.62	.61	.64	.000	1.5	.001	.000	.17	.000
treasury	.71	.71	.72	.000	1.9	.001	.000	.23	.000
wineRed	.90	.90	.93	.001	4.5	.002	.000	.51	.000
wineWhite	3.32	3.24	3.25	.002	50.6	.017	.000	5.82	.000
wizmir	1.05	1.06	1.06	.001	3.6	.001	.000	.40	.000
Mean	3.39	3.42	3.44	.002	73.1	.004	.000	8.35	.000
Mean rank	1.73	2.03	2.24	1.03	3.00	1.97	1.00	3.00	2.00

Table 4. Correlation between difficulty and test error

Dataset \ Technique	Correlation	
	M2	M3
abalone	.360	.372
anacalt	.853	.825
bank8fh	.172	.300
bank8fm	.404	.498
bank8nh	.196	.272
bank8nm	.602	.670
boston	.361	.429
comp	.443	.346
concreate	.352	.450
cooling	.821	.649
deltaA	.414	.425
deltaE	.176	.204
friedm	.348	.040
heating	.701	.628
istanbul	.072	.129
kin8fh	.221	.226
kin8fm	.557	.272
kin8nh	.248	.224
kin8nm	.468	.317
laser	.571	.695
mg	.668	.764
mortage	.667	.652
plastic	-.082	-.082
puma8fh	.264	.298
puma8fm	.234	.274
puma8nh	.241	.345
puma8nm	.222	.256
quakes	.133	.156
stock	.397	.348
treasury	.713	.594
wineRed	.238	.423
wineWhite	.257	.435
wizmir	.193	.214
Mean	.378	.383
Mean rank	1.67	1.33

coefficient of the two for each dataset, also produces the smallest average prediction region for 21 out of 33 cases. The probability of observing this (or a larger) number is only 0.081 if the resulting region size would be independent of this correlation.

5 Concluding Remarks

In this paper, we have presented a large-scale empirical evaluation of conformal regression approaches using random forests with out-of-bag calibration. We have compared a variance-based nonconformity measure, which has previously not been evaluated in this context, to a standard (non-normalized) nonconformity measure as well as to one measure based on k-nearest neighbors, which previously was found to give state-of-the-art performance. The experimental results in this study show that both the nearest-neighbor-based and the variance-based measures significantly outperform the non-normalized measure, while no significant difference in efficiency between the two normalized approaches is observed. Moreover, the evaluation shows that state-of-the-art performance is achieved by the variance-based measure at a computational cost that is several orders of magnitude lower than when employing the nearest-neighbor-based nonconformity measure.

There are several possible directions for future research. One direction concerns refining the rather straightforward difficulty estimate further, e.g., by not only considering variance of the ensemble member predictions, but also estimates of uncertainty for the individual predictions. Other directions for future research include investigating ways of combining several different difficulty estimates and evaluating the alternative nonconformity measures for other ensemble approaches for which out-of-bag estimates can be obtained.

Acknowledgments. This work was supported by the Swedish Foundation for Strategic Research through the project High-Performance Data Mining for Drug Effect Detection (IIS11-0053), the Vinnova program for Strategic Vehicle Research and Innovation (FFI)-Transport Efficiency, and the Knowledge Foundation through the project Data Analytics for Research and Development (20150185).

References

1. Bache, K., Lichman, M.: UCI machine learning repository (2013). http://archive.ics.uci.edu/ml
2. Breiman, L.: Bagging predictors. Mach. Learn. **24**(2), 123–140 (1996)
3. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001)
4. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. **7**, 1–30 (2006)
5. Gammerman, A., Vovk, V., Vapnik, V.: Learning by transduction. In: Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, pp. 148–155. Morgan Kaufmann (1998)

6. Johansson, U., Boström, H., Löfström, T., Linusson, H.: Regression conformal prediction with random forests. Mach. Learn. **97**(1–2), 155–176 (2014)
7. Löfström, T., Johansson, U., Boström, H.: Effective utilization of data in inductive conformal prediction. In: The 2013 International Joint Conference on Neural Networks (IJCNN). IEEE (2013)
8. Papadopoulos, H.: Inductive conformal prediction: theory and application to neural networks. Tools Artif. Intell. **18**(2), 315–330 (2008)
9. Papadopoulos, H., Haralambous, H.: Reliable prediction intervals with regression neural networks. Neural Netw. **24**(8), 842–851 (2011)
10. Papadopoulos, H., Proedrou, K., Vovk, V., Gammerman, A.J.: Inductive confidence machines for regression. In: Elomaa, T., Mannila, H., Toivonen, H. (eds.) ECML 2002. LNCS (LNAI), vol. 2430, pp. 345–356. Springer, Heidelberg (2002)
11. Papadopoulos, H., Vovk, V., Gammerman, A.: Regression conformal prediction with nearest neighbours. J. Artif. Intell. Res. **40**(1), 815–840 (2011)
12. Rasmussen, C.E., Neal, R.M., Hinton, G., van Camp, D., Revow, M., Ghahramani, Z., Kustra, R., Tibshirani, R.: Delve data for evaluating learning in valid experiments (1996). www.cs.toronto.edu/delve
13. Vovk, V., Gammerman, A., Shafer, G.: Algorithmic Learning in a Random World. Springer, Heidelberg (2006)