Normalized nonconformity measures for regression Conformal Prediction

Article · February 2008

CITATIONS
19

READS
717

3 authors, including:

Harris Papadopoulos
Frederick University
89 PUBLICATIONS **1,601** CITATIONS

SEE PROFILE

Alex Gammerman
Royal Holloway, University of London
105 PUBLICATIONS **5,456** CITATIONS

SEE PROFILE

NORMALIZED NONCONFORMITY MEASURES FOR REGRESSION CONFORMAL PREDICTION

Harris Papadopoulos
Computer Science and Engineering Department
Frederick University Cyprus
7 Y. Frederickou St., Palouriotisa,
Nicosia 1036, Cyprus
email: H.Papadopoulos@fit.ac.cy

Alex Gammerman and Volodya Vovk
Department of Computer Science,
Royal Holloway, University of London,
Egham Hill, Egham,
Surrey TW20 0EX, England
email: {Alex, Vovk}@cs.rhul.ac.uk

ABSTRACT
In this paper we apply Conformal Prediction (CP) to the k-Nearest Neighbours Regression (k-NNR) algorithm and propose a way of extending the typical nonconformity measure used for regression so far. Unlike traditional regression methods which produce point predictions, Conformal Predictors output predictive regions that satisfy a given confidence level. When the regular regression nonconformity measure is used the resulting predictive regions have more or less the same width for all examples in the test set. However, it would be more natural for the size of the regions to vary according to how difficult to predict each example is. We define two new nonconformity measures, which produce predictive regions of variable width depending on the expected accuracy of the algorithm on each example. As a consequence, the resulting predictive regions are in most cases much tighter than those produced by the simple regression measure.

KEY WORDS
Machine Learning, Conformal Prediction, Regression, Nearest Neighbours, Nonconformity Measure.

1 Introduction

A drawback of traditional machine learning algorithms is that they do not associate their predictions with confidence information, instead they only output *simple predictions*. However, some kind of confidence information about predictions is highly desirable in many risk-sensitive applications, such as those used for medical diagnosis or financial analysis.

There are some machine learning theories that produce confidence information. One can apply the theory of Probably Approximately Correct learning (PAC theory) to an algorithm in order to obtain upper bounds on the probability of its error with respect to some confidence level. The bounds produced by PAC theory though, will be very weak unless the data set to which the algorithm is being applied is particularly clean, which is rarely the case. For more details see [6], which demonstrates the crudeness of PAC bounds by applying a bound by Littlestone and Warmuth (found in [1], Theorem 4.25 and 6.8) to the USPS data set.

Another way to obtaining confidence information is by the use of Bayesian methods, which complement individual predictions with probabilistic measures of their quality. Bayesian methods produce strong bounds, but they require some a priori assumptions about the distribution generating the data. If these assumptions are violated the outputs of these methods can become quite misleading. An experimental demonstration of this negative aspect of Bayesian methods can be found in [4].

A different approach to confidence prediction was suggested in [2] (and later greatly improved in [12]), where what we call in this paper "Conformal Prediction" (CP) was proposed. An overview of CP is presented in [3] and a thorough analysis can be found in [14]. Conformal Predictors are built on top of traditional machine learning algorithms and accompany each of their predictions with a measure of confidence. Unlike Bayesian methods, CPs do not require any further assumptions about the distribution of the data, other than that the data are independently and identically distributed (i.i.d.). Furthermore, in contrast to PAC methods, the confidence measures they produce are useful in practice. Different variants of CPs are presented in [12, 13, 5, 7, 8, 9, 10].

In order to apply CP to a traditional algorithm one has to develop a *nonconformity measure* based on that algorithm. This measure evaluates the difference of a new example from a bag of old examples. Nonconformity measures are constructed using as basis the traditional algorithm to which the CP method is being applied, called the *underlying algorithm* of the resulting Conformal Predictor. In effect nonconformity measures assess the degree to which the new example disagrees with the attribute-label relationship of the old examples, according to the underlying algorithm of the CP. It is worth to note that many different nonconformity measures can be constructed for each traditional algorithm and each of those measures defines a different CP.

In this paper we are only interested in the problem of regression. The first regression CPs were proposed in [5] and [7] based on the Ridge Regression algorithm; actually in [7] a modified version of CP was introduced called "Inductive Conformal Prediction" (ICP). As opposed to the conventional point predictions, the output of regression

CPs is a predictive region that satisfies a given confidence level.

The typical nonconformity measure used so far in the case of regression is the absolute difference $|y_i - \hat{y}_i|$, between the actual label y_i of the example i and the predicted label $\hat{y}_i$ of the underlying algorithm for that example, given the bag of old examples as training set. In this paper we propose two extensions to this nonconformity measure for the *k*-Nearest Neighbours Regression CP. Our definitions normalize the standard measure based on the expected accuracy of the underlying algorithm for each example, thing that makes the size of the resulting predictive regions vary accordingly. As a result, the predictive regions produced by our measures are in most cases tighter than those produced by the standard regression measure.

In the next section we discuss the general idea on which regression CPs are based. We then, in section 3, describe the *k*-Nearest Neighbours Regression CP using the typical regression nonconformity measure, while in section 4 we give our two new nonconformity measure definitions and explain the rationale behind them. Section 5 details the experimental results of our methods and compares the tightness of their predictive regions. Finally, section 6 gives our conclusions.

2 Conformal Prediction for Regression

In this section we briefly describe the idea behind CP in the case of regression; for a more detailed description see [14]. We are given a training set $(z_1, \ldots, z_l)$ of examples, where each $z_i \in Z$ is a pair (x_i, y_i); $x_i \in \mathbb{R}^d$ is the vector of attributes for example i and $y_i \in \mathbb{R}$ is the label of that example. We are also given a new unlabeled example x_{l+1} and our task is to state something about our confidence in different values $\tilde{y}$ for the label y_{l+1} of this example. As mentioned above our only assumption is that all (x_i, y_i), $i = 1, 2, \ldots$, are generated independently from the same probability distribution.

First let us give a formal definition of a nonconformity measure. A nonconformity measure is a family of functions $A_n : Z^{(n-1)} \times Z \to \mathbb{R}, n = 1, 2, \ldots$ (where $Z^{(n-1)}$ is the set of all bags of size $n-1$), which assign a numerical score

$$\alpha_i = A_n(\{z_1, \ldots, z_{i-1}, z_{i+1}, \ldots, z_n\}, z_i)$$

to each example z_i, indicating how different it is from the examples in the bag $\{z_1, \ldots, z_{i-1}, z_{i+1}, \ldots, z_n\}$. Note that we use the concept of a bag so as to formalize the requirement that the order in which examples (except z_i) appear should not have any impact on the non-conformity score α_i.

Now suppose we are interested in some particular guess $\tilde{y}$ for the label of x_{l+1}. Adding this new example $(x_{l+1}, \tilde{y})$ to our known data set $((x_1, y_1), \ldots, (x_l, y_l))$ gives the sequence

$$(z_1, \ldots, z_{l+1}) = ((x_1, y_1), \ldots, (x_{l+1}, \tilde{y})); \qquad (1)$$

notice that the only component of this sequence that was not given to us is the label $\tilde{y}$. We can now use a nonconformity measure A_{l+1} to compute the nonconformity score

$$\alpha_i = A_{l+1}(\{z_1, \ldots, z_{i-1}, z_{i+1}, \ldots, z_{l+1}\}, z_i)$$

of each example z_i, $i = 1, \ldots, l+1$ in (1). The nonconformity score α_{l+1} on its own does not really give us any information, it is just a numeric value. However, we can find out how unusual z_{l+1} is according to A_{l+1} by comparing α_{l+1} with all other nonconformity scores. This comparison can be performed with the function

$$p(\tilde{y}) = \frac{\#\{i = 1, \ldots, l+1 : \alpha_i \geq \alpha_{l+1}\}}{l+1} \qquad (2)$$

(we leave the dependence of the left-hand side on $z_1, \ldots, z_l, x_{l+1}$ implicit, but it should be always kept in mind). We call the output of this function, which lies between $1/n$ and 1, the p-value of $\tilde{y}$, as that is the only part of (1) we were not given. An important property of (2) is that $\forall \delta \in [0, 1]$ and for all probability distributions P on Z,

$$P^{l+1}\{((x_1, y_1), \ldots, (x_{l+1}, y_{l+1})) : p(y_{l+1}) \leq \delta\} \leq \delta; \qquad (3)$$

for a proof see [6]. This makes it a valid test of randomness wrt the i.i.d. model. As a result, if the p-value of a given label is under some very low threshold, say 0.05, this would mean that this label is highly unlikely as such sequences will only be generated at most 5% of the time by any i.i.d. process.

Assuming we could calculate the p-value of every possible label $\tilde{y}$, as described above, we would be able to exclude all labels that have a p-value under some very low threshold (or *significance level*) δ and have at most δ chance of being wrong. Consequently, given a confidence level $1 - \delta$ a regression conformal predictor outputs the set

$$\{\tilde{y} : p(\tilde{y}) > \delta\}. \qquad (4)$$

In other words the set of all labels that have a p-value greater than δ. Of course it would be impossible to explicitly calculate the p-value of every possible label $\tilde{y} \in \mathbb{R}$. In the next section we describe how one can compute the predictive region (4) efficiently for *k*-Nearest Neighbours Regression.

It should be noted that the p-values computed by any CP will always be valid in the sense of satisfying (3) regardless of the particular nonconformity measure definition it uses. The choice of nonconformity measure definition only affects the tightness of the predictive regions output by the CP, and consequently their usefulness. To demonstrate the influence of an inadequate nonconformity measure definition on the results of a CP, let us consider the case of a trivial definition that always returns the value of 1 for any given example. This will make all p-values equal to 1 and will result in the predictive region $\mathbb{R}$ regardless of the required confidence level. Although this region will always contain the true label of the example, it is useless as it does not provide us with any information.

3 *k*-Nearest Neighbours Regression CP

The k-Nearest Neighbours algorithms base their predictions on the k training examples that are nearest to the unlabeled example in question based on some distance measure, such as the Euclidean distance. More specifically, for an input vector x_{l+1} the k-Nearest Neighbours Regression (k-NNR) algorithm finds the k nearest training examples to x_{l+1} and outputs the average (in some cases the median is also used) of their labels as its prediction. A refined form of the method assigns a weight to each one of the k examples depending on their distance from x_{l+1}, these weights determine the contribution of each label to the calculation of its prediction; in other words it predicts the weighted average of their labels. It is also worth to mention that the performance of the Nearest Neighbours method can be enhanced by the use of a suitable distance measure or kernel for a specific data set.

Here we will consider the version of the k-NNR method which predicts the weighted average of the k nearest examples. In our experiments we used the Euclidean distance, which is the most commonly used distance measure. It will be easy to see that the use of a kernel function or of a different distance measure will not require any changes to our method.

As mentioned in section 1 in order to create any CP we need to define a nonconformity measure based on the underlying algorithm in question. First let us consider the nonconformity measure

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

where $\hat{y}_i$ is the prediction of k-NNR for x_i based on the examples (x_j, y_j), $j = 1, \ldots, i-1, i+1, \ldots, l+1$; recall from section 2 that y_{l+1} is the label $\tilde{y}$ being tried for the new example x_{l+1}.

Following [5] and [14] we calculate the nonconformity score α_i for each example i as $\alpha_i = |a_i + b_i\tilde{y}|$. To do this we define a_i and b_i as follows:

- a_{l+1} is minus the weighted average of the labels of the k nearest neighbours of x_{l+1} and $b_{l+1} = 1$;
- if $i \leq l$ and x_{l+1} is one of the k nearest neighbours of x_i, a_i is y_i minus the labels of the $k-1$ nearest neighbours of x_i from $(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_l)$ multiplied by their corresponding weights, and b_i is minus the weight of x_{l+1};
- if $i \leq l$ and x_{l+1} is not one of the k nearest neighbours of x_i, a_i is y_i minus the weighted average of the labels of the k nearest neighbours of x_i, and $b_i = 0$.

As a result $|a_i + b_i\tilde{y}| = |y_i - \hat{y}_i|$ for $i = 1, \ldots, l+1$. Notice that now each $\alpha_i = \alpha_i(\tilde{y})$ varies piecewise-linearly as we change $\tilde{y}$. Therefore the p-value $p(\tilde{y})$ (defined by (2)) corresponding to $\tilde{y}$ can only change at the points where $\alpha_i(\tilde{y}) - \alpha_{l+1}(\tilde{y})$ changes sign for some $i = 1, \ldots, l$. This means that instead of having to calculate the p-value of every possible $\tilde{y}$, we can calculate the set of points $\tilde{y}$ on the real line that have a p-value $p(\tilde{y})$ greater than the given significance level δ, leading us to a feasible prediction algorithm.

For each $i = 1, \ldots, l+1$, let

$$\begin{aligned} S_i &= \{\tilde{y} : \alpha_i(\tilde{y}) \geq \alpha_{l+1}(\tilde{y})\} \\ &= \{\tilde{y} : |a_i + b_i\tilde{y}| \geq |a_{l+1} + b_{l+1}\tilde{y}|\}. \end{aligned} \tag{6}$$

Each set S_i (always closed) will either be an interval, a ray, the union of two rays, the real line, or empty; it can also be a point, which is a special case of an interval. As we are interested in $|a_i + b_i\tilde{y}|$ we can assume that $b_i \geq 0$ for $i = 1, \ldots, l+1$ (if not we multiply both a_i and b_i by -1). If $b_i \neq b_{l+1}$, then $\alpha_i(\tilde{y})$ and $\alpha_{l+1}(\tilde{y})$ are equal at two points (which may coincide):

$$-\frac{a_i - a_{l+1}}{b_i - b_{l+1}} \quad \text{and} \quad -\frac{a_i + a_{l+1}}{b_i + b_{l+1}}\,; \tag{7}$$

in this case S_i is an interval (maybe a point) or the union of two rays. If $b_i = b_{l+1} \neq 0$, then $\alpha_i(\tilde{y}) = \alpha_{l+1}(\tilde{y})$ at just one point:

$$-\frac{a_i + a_{l+1}}{2b_i}, \tag{8}$$

and S_i is a ray, unless $a_i = a_{l+1}$ in which case S_i is the real line. If $b_i = b_{l+1} = 0$, then S_i is either empty or the real line.

To calculate the p-value $p(\tilde{y})$ for any potential label $\tilde{y}$ of the new example x_{l+1}, we count how many S_i include $\tilde{y}$ and divide by $l+1$,

$$p(\tilde{y}) = \frac{\#\{i = 1, \ldots, l+1 : \tilde{y} \in S_i\}}{l+1}.$$

As $\tilde{y}$ increases $p(\tilde{y})$ can only change at the points (7) and (8), so for any significance level δ we can find the set of $\tilde{y}$ for which $p(\tilde{y}) > \delta$ as the union of finitely many intervals and rays. Algorithm 1 implements a slightly modified version of this idea. It creates a list of the points (7) and (8), sorts it in ascending order obtaining $y_{(1)}, \ldots, y_{(m)}$, adds $y_{(0)} = -\infty$ to the beginning and $y_{(m+1)} = \infty$ to the end of this list, and then computes $N(j)$, the number of S_i which contain the interval $(y_{(j)}, y_{(j+1)})$, for $j = 0, \ldots, m$, and $M(j)$ the number of S_i which contain the point $y_{(j)}$, for $j = 1, \ldots, m$.

4 Two New Nonconformity Measures

As mentioned in our introduction we extend nonconformity measure (5) normalizing it with the expected accuracy of the underlying method. More specifically, based on the idea behind the nearest neighbours algorithm, the nearer an example is to its k nearest training examples (the ones used for deriving the prediction) the more accurate one would expect the prediction of the algorithm to be. Therefore, it is stranger for the prediction of an example to be very different from its actual label when that example is near its k nearest neighbours, than when it is further away from them.

Algorithm 1: k-NNR CP

Input: training set $((x_1, y_1), \ldots, (x_l, y_l))$, new example x_{l+1}, number of nearest neighbours k and significance level δ.

$P := \{\}$;
for $i = 1$ **to** $l + 1$ **do**
  Calculate a_i and b_i for example $z_i = (x_i, y_i)$;
  if $b_i < 0$ **then** $a_i := -a_i$ and $b_i := -b_i$;
  if $b_i \neq b_{l+1}$ **then** add (7) to P;
  if $b_i = b_{l+1} \neq 0$ **and** $a_i \neq a_{l+1}$ **then** add (8) to P;
end
Sort P in ascending order obtaining $y_{(1)}, \ldots, y_{(m)}$;
Add $y_{(0)} := -\infty$ and $y_{(m+1)} := \infty$ to P;
$N(j) := 0, j = 0, \ldots, m$;
$M(j) := 0, j = 1, \ldots, m$;
for $i = 1$ **to** $l + 1$ **do**
  if $S_i = \{\}$ *(see (6))* **then** Do nothing;
  else if S_i *contains only one point,* $S_i = \{y_{(j)}\}$ **then**
    $M(j) := M(j) + 1$;
  else if S_i *is an interval* $[y_{(j_1)}, y_{(j_2)}]$, $j_1 < j_2$ **then**
    $M(z) := M(z) + 1, z = j_1, \ldots, j_2$;
    $N(z) := N(z) + 1, z = j_1, \ldots, j_2 - 1$;
  else if S_i *is a ray* $(-\infty, y_{(j)}]$ **then**
    $M(z) := M(z) + 1, z = 1, \ldots, j$;
    $N(z) := N(z) + 1, z = 0, \ldots, j - 1$;
  else if S_i *is a ray* $[y_{(j)}, \infty)$ **then**
    $M(z) := M(z) + 1, z = j, \ldots, m$;
    $N(z) := N(z) + 1, z = j, \ldots, m$;
  else if S_i *is the union* $(-\infty, y_{(j_1)}] \cup [y_{(j_2)}, \infty)$ *of two rays,* $j_1 < j_2$ **then**
    $M(z) := M(z) + 1, z = 1, \ldots, j_1, j_2, \ldots, m$;
    $N(z) := N(z) + 1, z = 0, \ldots, j_1 - 1, j_2, \ldots, m$;
  else if S_i *is the real line* $(-\infty, \infty)$ **then**
    $M(z) := M(z) + 1, z = 1, \ldots, m$;
    $N(z) := N(z) + 1, z = 0, \ldots, m$;
end
Output: the predictive region
$\left(\cup_{j:\frac{N(j)}{l+1} > \delta} (y_{(j)}, y_{(j+1)})\right) \cup \{y_{(j)} : \frac{M(j)}{l+1} > \delta\}$.

In our definitions we denote the sum of the k smallest distances of an attribute vector x_i from the attribute vectors $x_j, j = 1, \ldots, i-1, i+1, \ldots, l+1$ of the corresponding training examples as λ_i. The higher this value λ_i the less accurate we expect the prediction $\hat{y}_i$ to be. A first attempt to taking this into account could be

$$\alpha_i = \left| \frac{y_i - \hat{y}_i}{\lambda_i} \right|. \tag{9}$$

However, this definition is very sensitive to changes of λ_i, which takes very small values, usually less than 1. Because of this we defined the nonconformity measures:

$$\alpha_i = \left| \frac{y_i - \hat{y}_i}{\gamma + \lambda_i} \right|, \tag{10}$$

and

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

Learning Algorithm	Mean Absolute Error		
	housing	auto-mpg	CPU
k-Nearest Neighbours	2.66	2.02	30.7
Instance-based learning	2.90	2.72	34.0
Linear regression	3.29	2.61	35.5
Linear reg. + Instances	2.45	2.37	30.0
Model trees	2.45	2.11	28.9
Model trees + Instances	2.32	2.18	28.1
Neural nets	2.29	2.02	28.7
Neural nets + Instances	2.23	2.06	29.0

Table 1. Performance comparison of k-NNR with traditional algorithms.

each of which uses a different approach to overcome this problem. The first, which includes (9) as a special case, gives us the ability to adjust its sensitivity to changes of λ_i with the parameter γ; i.e. increasing γ results in a less sensitive nonconformity measure. The second, on the other hand, uses the exponential function, which has a minimum value of 1, since our λ will always be positive, and climbs quickly as λ increases. As a result, this measure is more sensitive to changes when λ is big, which indicates that an example is unusually far from the training examples. Again we can adjust the sensitivity of the measure with γ; in this case a bigger γ results in a more sensitive measure.

In order to calculate the nonconformity scores for (10) and (11) as $\alpha_i = |a_i + b_i \tilde{y}|$ we compute the a_i and b_i as defined in section 3 and then we divide both by $(\gamma + \lambda_i)$ for nonconformity measure (10) and by $\exp(\gamma \lambda_i)$ for nonconformity measure (11).

5 Experimental Results

Our methods were tested on three benchmark data sets from the UCI and DELVE repositories:

- Boston Housing, which lists the median house prices for 506 different areas of Boston MA in $1000s. Each area is described by 13 attributes such as pollution and crime rate.
- Auto-mpg, which concerns fuel consumption in miles per gallon (mpg). The original data set consisted of 398 examples out of which 6 were missing attribute values and were discarded. Furthermore, one attribute was also discarded as it was a unique identifying string. The remaining data set had 7 attributes such as engine capacity and number of cylinders.
- CPU Performance (or Machine), which lists the relative performance of 209 CPUs based on six continuous and 1 multivalued discrete attribute (the vendor). Two examples of the continuous attributes are cache memory size and cycle time.

Before our experiments the attributes of all data sets were normalized to a minimum value of 0 and a maximum

Data Set	Nonconformity Measure	Median Width			Percentage of label range (%)			Percentage outside predictive regions (%)		
		90%	95%	99%	90%	95%	99%	90%	95%	99%
Boston Housing	(5)	11.112	16.196	33.069	24.69	35.99	73.49	10.67	5.53	1.19
	(10) with $\gamma = 1$	10.333	13.916	26.019	22.96	30.92	57.82	10.28	5.53	1.19
	(10) with $\gamma = 0.5$	10.266	13.411	23.058	22.81	29.80	51.24	10.28	5.53	1.19
	(11) with $\gamma = 1$	10.128	13.513	24.450	22.51	30.03	54.33	10.28	5.14	1.19
Auto-mpg	(5)	8.897	11.244	20.490	23.66	29.90	54.49	9.69	4.85	1.02
	(10) with $\gamma = 1$	8.290	10.902	19.953	22.05	28.99	53.07	9.44	4.85	1.02
	(10) with $\gamma = 0.5$	7.958	10.923	19.921	21.16	29.05	52.98	9.44	5.10	1.02
	(11) with $\gamma = 1$	8.059	10.813	19.908	21.43	28.76	52.95	9.44	4.85	1.02
CPU Performance	(5)	168.10	225.65	1266.66	14.69	19.72	110.72	9.09	4.31	0.48
	(10) with $\gamma = 1$	135.92	174.44	339.46	11.88	15.25	29.67	9.57	3.83	0.48
	(10) with $\gamma = 0.5$	113.27	153.65	211.57	9.90	13.43	18.49	9.09	3.35	0.48
	(11) with $\gamma = 1$	119.79	154.70	221.47	10.47	13.52	19.36	9.09	3.83	0.00

Table 2. The tightness and reliability results of our methods on the three data sets.

value of 1. Following [11], each data set was tested using a 10 fold cross-validation process. Each data set was split randomly into 10 parts of almost equal size and our tests were repeated 10 times, each time using one of the 10 parts as the test set and the remaining 9 as the training set. For determining the number k of nearest neighbours that was used for each data set, one third of its training set was held-out as a validation set and the base algorithm was tested on that set with different k, using the other two thirds for training. The number k that gave the smallest mean absolute error was selected.

In table 1 we compare the performance of the base *k*-NNR method with the results reported in [11] for the three data sets. Of course our methods do not produce point predictions so these results are only listed to show that the performance of their underlying algorithm is comparable to that of other traditional methods.

All our experiments were repeated using each of the three nonconformity measures. For nonconformity measure (10) we considered both $\gamma = 1$ and $\gamma = 0.5$, while for nonconformity measure (11) we used $\gamma = 1$. Since our methods output predictive regions instead of point predictions, our main aim is to check the tightness of these regions. The first part of table 2 reports the median values of their sizes for the 99% 95% and 90% confidence levels. In the second part of the table each of these values is translated into a percentage of the whole range of possible labels for the corresponding data set. The reason for which we chose the median value instead of the mean is that it is more robust, i.e. if a few of the regions are extreme (either very large or very small) due to noise or over-fitting, the average will be affected while the median will remain unchanged.

In the third and final part of table 2 we check the reliability of the obtained predictive regions. This is done by reporting the percentage of examples for which the true label is not inside the region output by our method. In effect this checks empirically the validity of our predictive regions. The percentages reported here are very close to the required significance levels and do not change by much for the different nonconformity measures.

Figure 1 complements the information detailed in table 2 by displaying boxplots which show the median, upper and lower quartiles, and 10th and 90th percentiles of the predictive region widths produced for each data set. Each chart is divided into three parts, separating the three confidence levels we consider, and each part contains a boxplot for every nonconformity measure used.

Both table 2 and figure 1 show the relatively big improvement in predictive region tightness that our new nonconformity measures achieve as compared to the standard regression measure. Furthermore, the figures show that the widths of the regions produced by the new measures cover a much bigger range of sizes.

6 Conclusions

We have presented the *k*-NNR CP and have defined two new nonconformity measures for this method. Our experimental results show that the predictive regions produced by our method are relatively tight and consequently useful in practice. Furthermore, the two new nonconformity measures we defined produce regions of varying widths based on the expected accuracy of each example. This results in much tighter regions, for most examples, than those produced by the regular regression measure. Finally, our empirical reliability results confirm that our method can be used for obtaining reliable predictions.

Acknowledgements

This work was supported by the Cyprus Research Promotion Foundation through research contract PLHRO/0506/22 ("Development of New Conformal Prediction Methods with Applications in Medical Diagnosis").

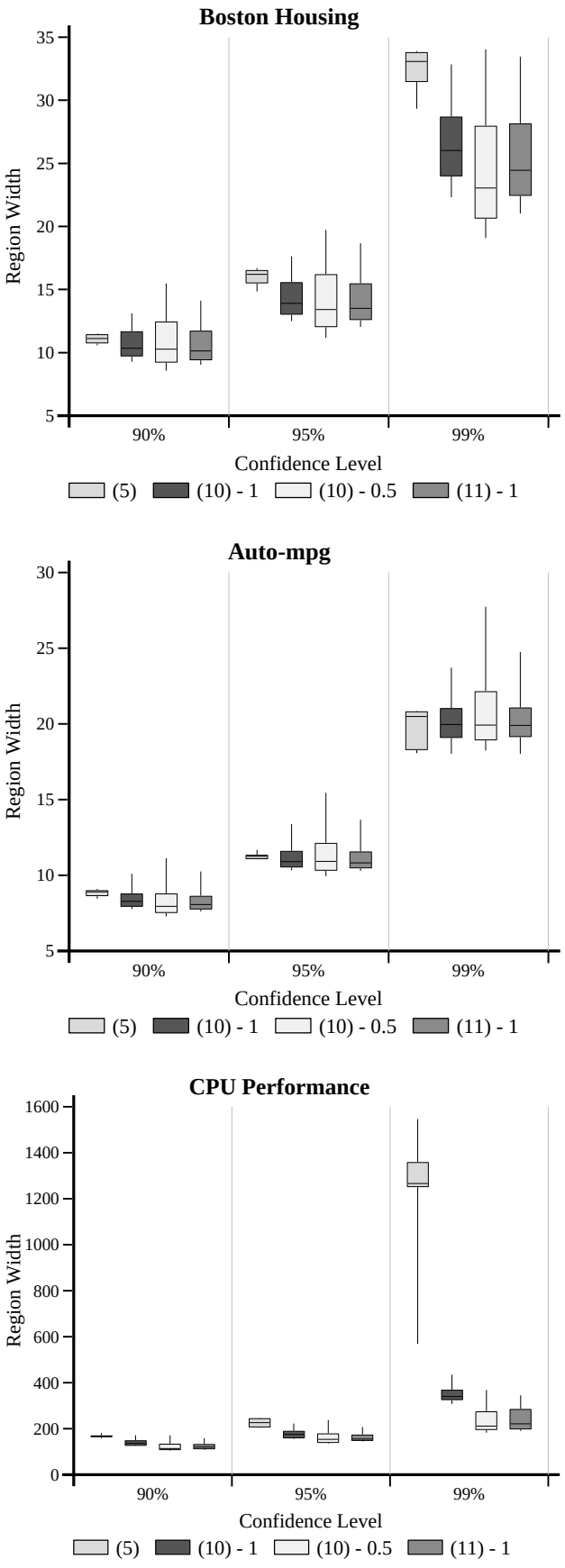

Figure 1. Distribution of predictive region widths.

References

[1] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Methods*. Cambridge University Press, Cambridge, 2000.

[2] A. Gammerman, V. Vapnik, and V. Vovk. Learning by transduction. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pages 148–156, San Francisco, CA, 1998. Morgan Kaufmann.

[3] A. Gammerman and V. Vovk. Hedging predictions in machine learning: The second *computer journal* lecture. *The Computer Journal*, 50(2):151–163, 2007.

[4] T. Melluish, C. Saunders, I. Nouretdinov, and V. Vovk. Comparing the Bayes and Typicalness frameworks. In *Proceedings of the 12th European Conference on Machine Learning (ECML'01)*, volume 2167 of *Lecture Notes in Computer Science*, pages 360–371. Springer, 2001.

[5] I. Nouretdinov, T. Melluish, and V. Vovk. Ridge regression confidence machine. In *Proceedings of the 18th International Conference on Machine Learning (ICML'01)*, pages 385–392, San Francisco, CA, 2001. Morgan Kaufmann.

[6] I. Nouretdinov, V. Vovk, M. V. Vyugin, and A. Gammerman. Pattern recognition and density estimation under the general i.i.d. assumption. In *Proceedings of the 14th Annual Conference on Computational Learning Theory (COLT'01) and 5th European Conference on Computational Learning Theory (EuroCOLT'01)*, volume 2111 of *Lecture Notes in Computer Science*, pages 337–353. Springer, 2001.

[7] H. Papadopoulos, K. Proedrou, V. Vovk, and A. Gammerman. Inductive confidence machines for regression. In *Proceedings of the 13th European Conference on Machine Learning (ECML'02)*, volume 2430 of *Lecture Notes in Computer Science*, pages 345–356. Springer, 2002.

[8] H. Papadopoulos, V. Vovk, and A. Gammerman. Qualified predictions for large data sets in the case of pattern recognition. In *Proceedings of the 2002 International Conference on Machine Learning and Applications (ICMLA'02)*, pages 159–163. CSREA Press, 2002.

[9] H. Papadopoulos, V. Vovk, and A. Gammerman. Conformal prediction with neural networks. In *Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'07)*, volume 2, pages 388–395. IEEE Computer Society, 2007.

[10] K. Proedrou, I. Nouretdinov, V. Vovk, and A. Gammerman. Transductive confidence machines for pattern recognition. In *Proceedings of the 13th European Conference on Machine Learning (ECML'02)*, volume 2430 of *Lecture Notes in Computer Science*, pages 381–390. Springer, 2002.

[11] J. R. Quinlan. Combining instance-based and model-based learning. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 236–243, Amherst, Massachusetts, 1993. Morgan Kaufmann.

[12] C. Saunders, A. Gammerman, and V. Vovk. Transduction with confidence and credibility. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, volume 2, pages 722–726. Morgan Kaufmann, 1999.

[13] C. Saunders, A. Gammerman, and V. Vovk. Computationally efficient transductive machines. In *Proceedings of the Eleventh International Conference on Algorithmic Learning Theory (ALT'00)*, volume 1968 of *Lecture Notes in Artificial Intelligence*, pages 325–333, Berlin, 2000. Springer.

[14] V. Vovk, A. Gammerman, and G. Shafer. *Algorithmic Learning in a Random World*. Springer, New York, 2005.