Nearest Neighbor Regression in the Presence of Bad Hubs

Krisztian Buza[a], Alexandros Nanopoulos[b], Gábor Nagy[c]

[a]*BioIntelligence Lab, Genomic Medicine and Rare Disorders, Semmelweis University buza@biointelligence.hu*
[b]*University of Eichstätt-Ingolstadt, alexandros.nanopoulos@ku.de*
[c]*Budapest University of Technology and Economics, nagy.gabor.i@gmail.com*

Abstract

Prediction on a numeric scale, i.e., regression, is one of the most prominent machine learning tasks with various applications in finance, medicine, social and natural sciences. Due to its simplicity, theoretical performance guarantees and successful real-world applications, one of the most popular regression techniques is the k nearest neighbor regression. However, k nearest neighbor approaches are affected by the presence of *bad hubs*, a recently observed phenomenon according to which some of the instances are similar to surprisingly many other instances and have a detrimental effect on the overall prediction performance. This paper is the first to study bad hubs in context of regression. We propose hubness-aware nearest neighbor regression schemes. We evaluate our approaches on publicly available real-world datasets from various domains. Our results show that the proposed approaches outperform various other regressions schemes such as kNN regression, regression trees and neural networks. We also evaluate the proposed approaches in the presence of label noise because tolerance to noise is one of the most relevant aspects from the point of view of real-world applications. In particular, we perform experiments under the assumption of conventional Gaussian label noise and an adapted version of the recently proposed hubness-proportional random label noise.

Keywords: nearest neighbor regression, hubs

1. Introduction

Regression, i.e., prediction of a continuous target variable from a set of observations, is one of the most prominent machine learning tasks with various applications in engineering, finance, industry and medicine, see e.g. [1], [2], [3], [4], [5]. Various regression techniques have been developed in the last decades ranging from simple linear and polynomial regression to more complex models, such as neural networks [6], [7] and support vector regression [8].

In many cases, not only the nominal dimensionality of the data is high, but the same is true for the number of *meaningful* (or *intrinsic*) dimensions, although the later may vary from instance to instance: for example, the medical records of a patient p may involve the results of different examinations than the records of another patient p' (who may suffer from different diseases than p). Therefore, making use of such data is not only difficult because of its size, but also due to its complexity: without loss of essential information, calculating the distance or similarity between instances may already be rather challenging, while finding a reasonable vector representation of the data may be even more difficult. On the other hand, there is an ever-growing interest in using such semi-structured data for prediction, which involves prediction on a numeric scale, i.e., regression. Consequently, in this paper we focus on regression techniques that only assume the presence of an appropriate distance or similarity measure which *may* or *may not* be based on the vector representation of the instances.

Despite the aforementioned variety of regression schemes, one of the most popular techniques is the nearest neighbor regression. While being intuitive, nearest neighbor regression is well-understood from the point of view of theory, see e.g. [9], [10] and [11] and the references therein for an overview of the most important theoretical results regarding the performance of nearest neighbor regression. These theoretical results are also justified by empirical studies: for example, in their recent paper, Stensbo-Smidt et al. found that nearest neighbor regression outperforms model-based prediction of star formation rates [12], while Hu et al. showed that a k-nearest neighbor regression based model is able to estimate the capacity of lithium-ion batteries [3].

We point out that most of the conventional regression schemes were developed for vector data, i.e., under the assumption that the data can be organized into a data table with well-defined dimensionality, whereas in the aforementioned cases this assumption may be violated. However, nearest neighbor-models only require a distance or similarity between the instances, which may be much simpler to define than finding a suitable vector representation of the data, see e.g. edit distances for time series, genetic sequences or texts, such as dynamic time warping [13], [14], Smith-Watermann distance [15] or Levenshtein distance [16]. These distance measures work directly on the "raw" data (i.e., time series, genetic sequences or texts respectively) without an intermediate vector representation.

Machine learning in high dimensional data spaces is particularly challenging due to the phenomena known under the umbrella of the *curse of dimensionality.* One of the recently explored aspects of the curse is the emergence of bad hubs, see e.g. [17], [18], [19], [20], [21]. Informally, hubs are instances that are similar to a surprisingly high amount of other instances. Unfortunately, some of the hubs are bad in the sort of sense that they may mislead classification algorithms. While bad hubs are well-studied in case of classification [22], instance selection [23] and clustering [24], in context of regression problems bad hubs have not been described yet. Providing an analysis of the presence of bad hubs in regression problems is not trivial because the original definition of bad hubs assumes discrete class labels, however, in case of regression problems, the labels are continuous. Therefore, in order to study bad hubs in context of regression, we need a novel approach.

In this paper, we focus on nearest neighbor regression and study the presence of bad hubs in context of regression problems. Motivated by these observations, we propose hubness-aware nearest neighbor regression schemes. Subsequently, we evaluate our approach on publicly available real-world datasets from various domains: prediction of yields on the stock market, assessment of the severity of Parkinson's disease, estimation of the area of forest fires, prediction of the number of comments that a blog post will receive and assessment of wine quality. Our experimental results show that our approach is favorable in all these domains. Additionally, we evaluate the proposed approaches in the presence of label noise because, on the one hand, tolerance to noise is one of the most relevant aspects from the point of view of real-world applications, on the other hand, the selection of appropriate noise models allow us to simulate the increased presence of bad hubs. In particular, we perform experiments under the assumption of two types of noise: we consider conventional Gaussian label noise and an adapted version of the recently proposed hubness-proportional random label noise [25]. This adaptation is one of the minor contributions of the paper and it is necessary because hubness-proportional random label noise was originally introduced for classification problems. According to the best of our knowledge, this is the first paper that studies the presence of bad hubs in context of regression problems, and this is the first paper that proposes hubness-aware regression schemes and evaluates them both on real-world datasets and under various noise models.

2. Definitions and notations

A dataset $\mathcal{D}$ containing n instances is given. Instances are denoted by $x_i, 1 \leq i \leq n$. For each instance $x_i \in \mathcal{D}$, the value of the continuous target is given and it is denoted by $y(x_i)$. We say that $y(x_i)$ is the *label* of instance x_i and $\mathcal{D}$ is the training dataset. With regression we mean the task of predicting (estimating) the label of an instance $x' \notin \mathcal{D}$.

We propose a regression approach that is independent of the representation of the instances, the only requirement is that distances can be defined between the instances. Therefore, we use $d(x_i, x_j)$ to denote the distance between two instances x_i and x_j.

Assume that we want to predict the label of an instance $x' \notin \mathcal{D}$. Nearest neighbor regression determines the k nearest neighbors of x', i.e., a subset $\mathcal{N}_k^{\mathcal{D}}(x')$ of $\mathcal{D}$ so that

$$|\mathcal{N}_k^{\mathcal{D}}(x')| = k \tag{1}$$

and

$$\max_{x \in \mathcal{N}_k^{\mathcal{D}}(x')} d(x', x) \leq \min_{x \in \mathcal{D} \setminus \mathcal{N}_k^{\mathcal{D}}(x')} d(x', x). \tag{2}$$

We may omit the upper index $\mathcal{D}$, whenever it is unambiguous in which dataset we search for the nearest neighbors of x'. Nearest neighbor regression [26],[27] estimates the value of the target as the average of the labels of the nearest neighbors:

$$\hat{y}(x') = \frac{1}{k} \sum_{x_j \in \mathcal{N}_k(x')} y(x_j). \tag{3}$$

3. Bad Hubs in Regression Problems

We note that the k nearest neighbor relationship is asymmetric: while each instance $x \in \mathcal{D}$ has k nearest neighbors, an instance $x' \in \mathcal{D}$ does not necessarily appear k-times as one of the k nearest neighbors of other instances. This is illustrated in Fig. 1 for $k = 1$. In order to keep the example simple, we consider two-dimensional vector data, therefore, instances correspond to points of the plane. In Fig. 1, instances are denoted by circles. There is a directed edge from each instance to its first nearest neighbor. While each instance has *exactly one* first nearest neighbor, i.e., the number of *outgoing* edges is exactly one for each instance; how many times an instance appears as the first nearest neighbor of *other* instances, i.e., the number of *incoming* edges, is not necessarily one. As one can see, some of the instances never appear as nearest neighbors of others and there is an instance that appears as the first nearest neighbor of three other instances: the integer number next to each instance shows how many times it appears as the first nearest neighbor of others.

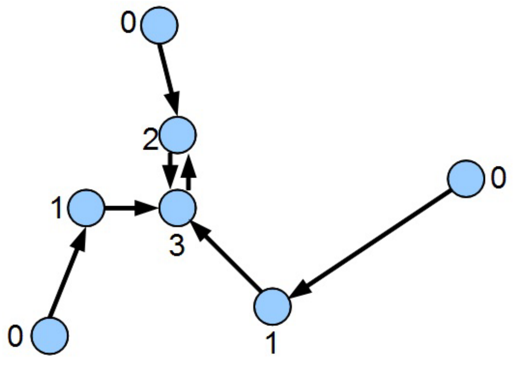


Figure 1: The nearest neighbor relationship is asymmetric. On the one hand, some instances never appear as the first nearest neighbor of other instances; on the other hand, there are some instances that appear frequently as the first nearest neighbor of other instances.

Generally, we use $N_k(x)$ to denote how many times the instance $x \in \mathcal{D}$ appears as one of the k nearest neighbors of other instances of $\mathcal{D}$. It is easy to see that the expected value of $N_k(x)$ is $E[N_k(x)] = k$, however, the actual value of $N_k(x)$ varies from instance to instance. While considering k nearest neighbor models, $N_k(x)$ can be seen as the measure of how influential is the instance x. As it was shown in previous works, see e.g. [17], [18], [23], in many cases, the distribution of $N_k(x)$ is substantially skewed to the right, i.e., there are a few instances with extraordinarily high $N_k(x)$ values. Usually, instances having surprisingly high $N_k(x)$ are called *hubs*, while instance with exceptionally low $N_k(x)$ are called *anti-hubs*. More precisely, we say that an instance x is a hub, if $N_k(x) > 2k$; while an instance x is an *anti-hub* if $N_k(x) = 0$. The phenomenon that $N_k(x)$ is skewed is called *hubness* and it is often quantified by the third standardized moment (skewness) of the distribution of $N_k(x)$.

In case of classification, we say that an instance x is a *bad k nearest neighbor* of another instance x' if x is one of the k-nearest neighbors of x' and the both instances have different class labels. Consequently, in case of classification, *bad k-occurrence* $BN_k(x)$ of an instance x was introduced to measure how many times an instance x appears as bad nearest neighbor of other instances. Similarly to the distribution of $N_k(x)$, the distribution of $BN_k(x)$ was shown to be substantially skewed in case of high-dimensional data.

Similar observations can be made for high-dimensional data associated with numerical prediction tasks. As an example, in the left of Figure 2 we show the distribution of $N_k(x)$ for the Financial Tweets Data using $k = 10$ and the Euclidean distance. The dataset is described in Section 5.1.1 in more detail. In Figure 2 horizontal axis corresponds to $N_{10}(x)$ while the height of the column shows how many instances have that particular value of $N_{10}(x)$.

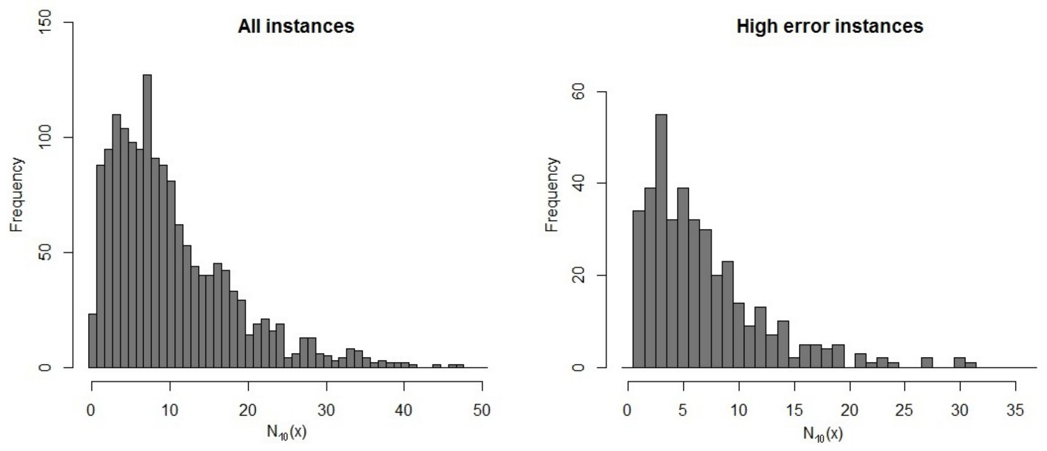


Figure 2: The distribution of $N_{10}(x)$ for all the instances (in the left) and high error instances (in the right) of the Financial Tweets dataset. As one can see, some of the high error instances appear as nearest neighbors of many other instances, i.e., there are bad hubs in the data.

Next, we aim to describe those instances that are potentially most harmful for nearest neighbor regression. On the one hand, instances that rarely (or even never) appear as nearest neighbors of other instances have low (or no) impact on the predicted labels. On the other hand, the closer is the label of an instance x to the labels of those instances for which x appears as one of their nearest neighbors, the less harmful is x. Therefore, an instance x

may be harmful to k nearest neighbor regression, if (i) x appear as nearest neighbors of many other instances, i.e., $N_k(x)$ is relatively large, *and* (ii) the label of x is substantially different from the label of those instances that have x as one of their nearest neighbors.

In order to show that there are misleading instances in real datasets, we perform the following analysis on the previously mentioned Financial Tweets dataset. From the prediction tasks associated with the data, we considered the yield prediction for the next day, i.e., yields on the next day were used as instance labels. For each instance x, as $error(x)$, we calculate the average absolute difference between the label of x and the labels of those instances that have x as one of their nearest neighbors. Formally, let

$$\mathcal{I}_x = \{x_j | x \in \mathcal{N}_k(x_j)\}, \tag{4}$$

then

$$error(x) = \frac{1}{|\mathcal{I}_x|} \sum_{x_i \in \mathcal{I}_x} |y(x) - y(x_i)|. \tag{5}$$

After calculating the above error for each instance, we ordered the instances according to their errors and selected 25% of the instances having the *highest error.* We call these instances *high error instances.* The distribution of $N_{10}(x)$ for high error instances in shown in the right of Figure 2. As one can see, similarly to the previous distribution, the distribution of $N_{10}(x)$ for high error instances is notably skewed as well. More importantly, some of the high error instances appear as nearest neighbors of many other instances. In this paper, we define *bad hubs* as high error instances that appear as nearest neighbors of more than $2k$ instances. As one can see, there are bad hubs in the data.

The above observations suggest that a few high error instances may have substantial detrimental effect on the prediction performance. The mutual k nearest neighbor (MkNN or MNN) approach can be assen as an attempt to alleviate this problem [27]. While predicting the label of an instance x', MkNN takes only those neighbors x_j of x' into account that have x' as one of their k nearest neighbors, i.e., using

$$\mathcal{M}_k(x') = \{x_j \in \mathcal{N}_k^{\mathcal{D}}(x') : x' \in \mathcal{N}_k^{\mathcal{D} \cup \{x'\}}(x_j)\}, \tag{6}$$

the MkNN estimate is

$$\hat{y}^{(M)}(x') = \frac{1}{|\mathcal{M}_k(x')|} \sum_{x_j \in \mathcal{M}_k(x')} y(x_j). \tag{7}$$

"Without claiming that MNN estimates always outperform standard nearest neighbors estimates," [27] Guyader and Hengartner argued that the presence of bad hubs might not substantially affect the performance of MkNN. While we acknowledge this argumentation, we note that MkNN does not distinguish between good and bad hubs. This may be the reason why approaches taking the presence of bad hubs directly into account may outperform MkNN as we will show in the subsequents sections.

4. Hubness-aware regression schemes

4.1. Error-based weighting

For each instance x of the training dataset that appears as nearest neighbor at at least one other instance, we can calculate $error(x)$ according to Eq. (5), i.e., the average absolute difference between the label of x and the labels of those instances that have x as one of their k nearest neighbors. We assign weights to instances so that the higher is the error of an instance, the less is its influence. While in general there are many ways of doing that, here, we use the normalized error $err(x)$ of an instance x to define its weight $w(x)$:

$$err(x) = \frac{error(x) - \mu_{err}}{\sigma_{err}}, \tag{8}$$

$$w(x) = e^{-err(x)}, \tag{9}$$

where μ_{err} and σ_{err} denote the average and standard deviation of the error respectively. The above weights are assigned to all instances with $N_k(x) \geq 1$. Instances with $N_k(x) = 0$ are removed from the training set after calculating the weights. See also Section 4.4 for further discussion on this reduction of the training data.

Subsequently, we perform weighted k nearest neighbor regression, i.e., we estimate the label of a new instance x' as

$$\hat{y}^{(W)}(x') = \frac{\sum\limits_{x_j \in \mathcal{N}_k(x')} w(x_j) y(x_j)}{\sum\limits_{x_j \in \mathcal{N}_k(x')} w(x_j)}. \tag{10}$$

As we utilize error-based weighting of instances for k nearest neighbor regression, we call this approach EWkNN.

4.2. Error correction

Although the weighting scheme shown in the previous section allows to reduce the influence of high error instances, in EWkNN, each instance votes by its own, possibly misleading label. In this section we propose that each instance x uses the average of the labels of those instances that have x as one of their nearest neighbors. Formally, we can define the *corrected label* $y_c(x)$ of an instance x appearing as nearest neighbor at at least one other instance as

$$y_c(x) = \frac{1}{|\mathcal{I}_x|} \sum_{x_i \in \mathcal{I}_x} y(x), \tag{11}$$

where $\mathcal{I}_x$ denotes the set of instances that have x as one of their nearest neighbors, see Eq. (4) for the formal definition of $\mathcal{I}_x$.

The corrected labels are calculated for all instances with $N_k(x) \geq 1$. Instances with $N_k(x) = 0$ are removed from the training set after calculating the corrected labels. See also Section 4.4 for further discussion on this reduction of the training data.

Subsequently, we can use the corrected labels in nearest neighbor regression and estimate the label of a new instance x' as

$$\hat{y}^{(C)}(x') = \frac{1}{k} \sum_{x_j \in \mathcal{N}_k(x')} y_c(x_j). \tag{12}$$

As we use the corrected labels for prediction, we call this approach Error Correction-based k Nearest Neighbor regression, or simply ECkNN.

4.3. Error-based weighting and correction

We may combine the two previous ideas and use the corrected labels in a weighted regression schema so that the predicted label of a new instance x' is

$$\hat{y}^{(WC)}(x') = \frac{\sum\limits_{x_j \in \mathcal{N}_k(x')} w(x_j) y_c(x_j)}{\sum\limits_{x_j \in \mathcal{N}_k(x')} w(x_j)}. \tag{13}$$

We call this approach EWCkNN, error-based weighting and correction for k nearest neighbor regression.

4.4. Data Reduction

Under the assumption that the test data originates from the same distribution as the training data, instances which never appear as nearest neighbor of other instances are not expected to influence predictions of nearest neighbor regressors. Therefore, at the end of the training process of the proposed regressors, i.e., after calculation of the weights and/or error correction, as described in the previous sections, we remove instances with $N_k(x) = 0$ from the training data. The removal of the instances with $N_k(x) = 0$ reduces the computational costs of nearest neighbor regression too. For further hubness-aware instance selection techniques we refer the Reader to [23],[28].

4.5. Illustrative example

Using the example in Figure 3 we illustrate how the proposed regression approaches perform prediction. In Figure 3 training instances are denoted by circles. They are identified by the symbols $x_1 ... x_7$. The numeric value next to each instance shows its label. We aim at predicting the label of the test instance that is denoted by the triangle.

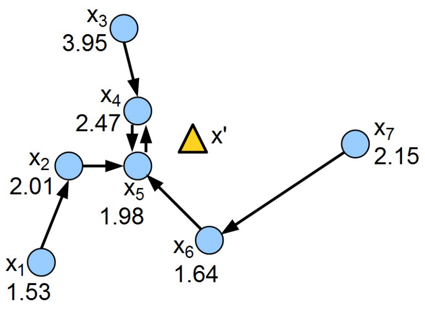


Figure 3: Example used to illustrate the proposed hubness-aware regression approaches. Training instances of the dataset are denoted by circles. We aim at predicting the label of the instance denoted by the triangle.

In order to keep the example simple, we use $k = 1$ to calculate $error(x)$, $w(x)$ and the corrected labels training instances that appear as nearest neighbor of at least one other training instance. However, in order to illustrate weighted nearest neighbor regression, we use $k' = 2$ nearest neighbors to predict the label of the test instance.

First, we illustrate EWkNN. We begin with calculating $error(x)$ of training instances that appear as nearest neighbor of at least one other training instance. These calculations are performed according to Eq. (5), therefore:

$error(x_2) = |2.01 - 1.53| = 0.48$ because x_2 appears as the nearest neighbor of x_1 only;

$$error(x_4) = \frac{1}{2}(|2.47 - 3.95| + |2.47 - 1.98|) = 0.985$$

because x_4 appears as nearest neighbor of the instances x_3 and x_5. Similarly,

$$error(x_5) = \frac{1}{3}(|1.98 - 2.01| + |1.98 - 2.47| + |1.98 - 1.64|) \approx 0.287,$$

$error(x_6) = |1.64 - 2.15| = 0.51$. The average and standard deviation of the error is $\mu_{err} = 0.566$ and $\sigma_{err} = 0.297$ respectively.

The $k' = 2$ nearest neighbors of the test instance are the instances x_4 and x_5. Therefore, we calculate the normalized error and the weights of these instances according to Equations (8) and (9) as follows:

$$err(x_4) = \frac{0.985 - 0.566}{0.297} = 1.411, \quad w(x_4) = e^{-1.411} = 0.244$$

$$err(x_5) = \frac{0.287 - 0.566}{0.297} = -0.939, \quad w(x_5) = e^{0.939} = 2.558.$$

The label predicted by EWkNN is the weighted average of the labels of the $k' = 2$ nearest neighbors of the test instance:

$$\hat{y}^{(W)}(x') = \frac{0.244 \cdot 2.47 + 2.558 \cdot 1.98}{0.244 + 2.558} = 2.023.$$

Next, we illustrate the proposed error correction approach. According to Eq.(11) the corrected labels of the training instance x_4 is the average of the labels of those instances that have x_4 as their nearest neighbor. In particular:

$$y_c(x_4) = \frac{1}{2}(3.95 + 1.98) = 2.965.$$

Similarly, the corrected label of x_5 is

$$y_c(x_5) = \frac{1}{3}(2.01 + 2.47 + 1.054) = 2.04.$$

The label predicted by ECkNN is

$$\hat{y}^{(C)}(x') = \frac{1}{2}(2.965 + 2.04) = 2.5025.$$

EWCkNN uses the weights calculated by EWkNN together with the corrected class labels, therefore the label predicted by EWCkNN is

$$\hat{y}^{(WC)}(x') = \frac{0.244 \cdot 2.965 + 2.558 \cdot 2.04}{0.244 + 2.558} = 2.121.$$

5. Experiments

In this section we present the results of our experimental evaluation of the proposed approaches, i.e., EWkNN, ECkNN and EWCkNN.

5.1. Datasets

We evaluated our approach on real-world data associated with different applications from various domains. We collected financial tweets and tried to predict the yield of particular stocks based on those tweets. Furthermore, in order to assist reproducibility of our experiments, we used publicly available real-world datasets associated with various prediction tasks, such as the assessment of the severity of Parkinson's disease, prediction of the area of forest fires, prediction of the number of comments that a blog post will receive and assessment of wine quality. The later datasets are available in the UCI Machine Learning repository [29]. The datasets used in this study are summarized in Table 1. The underlying applications are described in the subsequent sections in more detail.

Table 1: Overview of the datasets used in our study. For each dataset, we provide (i) the number of instances, (ii) its nominal dimensionality d, i.e., the number of features used for prediction, (iii) its intrinsic dimensionality d_0, (iv) the proportion of bad hubs, (v) a short description of the data and (vi) the regression task.

Dataset	#Inst.	d	d_0	%bad hubs	Short description	Prediction target
Financial Tweets (FT)	1565	65	17	10.5	Numeric descriptors of stocks extracted from short messages that appeared on twitter.	Yield in the future.
Parkinsons Telemonitoring	5875	16	6	10.1	Biomedical voice measurements.	UPDRS-score of patients
Forest Fires	517	10	7	4.7	Meteorological descriptors	Area of fire
Blog Feedback	60,021	280	20	9.4	Descriptors of blogs	Number of comments
Wine Quality (white)	4,898	11	5	9.3	Physiochemical features of wines	Quality of wines
Parkinson multisound	1,040	26	14	6.6	Parkinson speech dataset with multiple types of sound recordings	UPDRS-score of patients

For each dataset, the second and third column of Table 1 show the number of instances and number of features used for the prediction, i.e., the nominal dimensionality. Regarding the presence of hubs, as noted by Radovanović et al. [17], the *intrinsic* dimensionality of the data plays a more important role than its nominal dimensionality. Therefore, for each dataset we report its intrinsic dimensionality in the fourth column of Table 1. We estimated the intrinsic dimensionality as follows: for each dataset, we performed dimensionality reduction with PCA using the R statistical software[1] and checked how many principal components are required to keep 99 % of the total variance. As one can see, the intrinsic dimensionality is often much lower than the nominal dimensionality which is in accordance with the observations made by Radovanović et al. We note that Radovanović et al. reported that hubs are already present in datasets with intrinsic dimensionality around ten, which is also in accordance with our observations.

We quantified the presence of bad hubs as follows: we considered the high-error instances as described previously and checked how many of them appear as nearest neighbors of at least $2k$ instances. For the estimation of the amount of bad hubs we used $k = 3$. The amount of bad hubs (in percent of all the high error instances) is shown in the fifth column of Table 1.

5.1.1. Financial Tweets Dataset

In order to demonstrate that our approach may be useful for prediction tasks in the financial domain, we collected financial tweets and tried to predict the yield of particular stocks based on those tweets.

Each instance of the Financial Tweets dataset corresponds to one of 1565 stocks. In order to collect the data, we crawled tweets containing the symbols (few letter abbreviations) of prominent stocks for the time period of 25 days between the 13th May and the 6th June 2014. From the raw data we extracted 25 numeric features for each stock. Each of these features corresponds to a particular day and gives how many times the stock was mentioned on that day in the tweets. Additionally, for each business day we include the return and the volume of trade of the particular stock. We calculated three further features: the standard deviation of the returns, volumes of trade and how many times the stock was mentioned in the considered period of time. The associated regression task is to predict the yield (in percent) of each

[1]http://www.r-project.org

stock for the (i) next business day (i.e., the 9th of June), (ii) next week, (iii) next two weeks and (iv) next three weeks. We evaluated our approach using all the four targets, however, we considered the yield on the next business day as default target, i.e., whenever the target is not explicitly stated, we used the yield for the next business day as target.

5.1.2. Datasets related to Parkinson's Disease

"Parkinson's disease (...) is a degenerative disorder of the central nervous system."[2] It can cause neuropsychiatric disturbances which can range from mild to severe. This includes disorders of speech, cognition, mood, behaviour, and thought. [30] The Unified Parkinson's Disease Rating Scale (UPDRS) is used to assess the severity of the disease.

The Parkinsons Telemonitoring (TM) dataset [31, 32] "is composed of a range of biomedical voice measurements from 42 people with early-stage Parkinson's disease recruited to a six-month trial of a telemonitoring device for remote symptom progression monitoring."[3] We report results for using the 16 voice measures of the data to predict the total and motor UPDRS scores. We note that we considered the total UPDRS score as default target, i.e., whenever the target is not explicitly stated, we used the total UPDRS score.

We also used the *Parkinson Speech Dataset with Multiple Types of Sound Recordings* to which we refer as *MultiSound* for simplicity [33]. This data is associated with a similar prediction task, i.e., the prediction of the UPDRS score.

5.1.3. Further Datasets

The Forest Fires dataset contains 517 instances. The associated regression task is to predict the area of forest fires based on meteorological features like temperature, wind, rain or the month of the year [34]. Except the target, the dataset has 12 features. In order to keep our experiments uniform, for all the datasets we used the Euclidean distance. Therefore, out of the 12 features of the data, we only used the 10 numeric features and we ignored the remaining two nominal (string-valued) features.

The Blog Feedback dataset contains 60,021 instances and 281 features including the target variable. Each instance corresponds to a blog post.

[2]http://en.wikipedia.org/wiki/Parkinson%27s_disease

[3]http://archive.ics.uci.edu/ml/datasets/Parkinsons+Telemonitoring

Users may comment on blog posts, however, the number of comments that particular blog posts receive highly vary from post to post. For each blog post, the goal is to predict the number of comments in the upcoming 24 hours [35]. In order to simulate real-world scenarios in which predictions based on past data has to be made for the future, the data comes with temporal train and test splits.

In our experiments on the Wine Quality Dataset[4] [36], we aimed to predict the quality of white wines based on physiochemical features.

5.2. Experimental Protocol

We used the provided temporal train and test splits of the Blog Feedback dataset. As the other datasets did not have pre-defined train and test splits, we performed experiments according to the 10-fold cross-validation protocol on those datasets.

5.3. Compared Methods

As the proposed approaches, EWkNN, ECkNN and EWCkNN, are based on nearest neighbor regression, we used nearest neighbor regression (as described in Section 2) as primary baseline. Throughout the experimental results, nearest neighbor regression is denoted by kNN. We used the Euclidean distance in all the aforementioned regression approaches.

Additionally, in order to study if the proposed methods are competitive to other regression techniques, we run experiments using MkNN, linear regression, M5P regression trees, feed forward neural networks with one hidden layer and 5 and 10 hidden nodes respectively. We tried MkNN both with $k = 5$ and $k = 10$. As MkNN with $k = 10$ performed remarkably better than with $k = 10$, we decided to report results only for the case of $k = 10$ for MkNN. These techniques are denoted by MkNN, LinReg, M5P, Net-5 and Net-10 respectively. In our experiments, we used the Weka-implementations of linear regression, regression trees and neural networks [37].

5.4. Performance Metrics

We measured the performance of our approach and the baselines in terms of mean absolute error (MAE) and normalized mean absolute error (NMAE):

$$MAE = \frac{1}{|\mathcal{D}_{test}|} \sum_{x_j \in \mathcal{D}_{test}} |\hat{y}(x_j) - y(x_j)|,$$

[4]https://archive.ics.uci.edu/ml/datasets/Wine+Quality

$$NMAE = \frac{1}{|\mathcal{D}_{test}|} \sum_{x_j \in \mathcal{D}_{test}} \frac{|\hat{y}(x_j) - y(x_j)|}{|y(x_j)|},$$

where $\mathcal{D}_{test}$ and $|\mathcal{D}_{test}|$ denote the test set and its size respectively, $\hat{y}(x_j)$ denotes the label predicted by the regression model (i.e., one of kNN, EWkNN, ECkNN or EWCkNN, MkNN, LinReg, M5P, NeuralNet-5 or NeuralNet-10) for instance x_j, while $y(x_j)$ denotes the true label of instance x_j.

We used paired t-test at significance level of 0.05 to examine if EWkNN, ECkNN and EWCkNN statistically significantly outperform the baseline. For simplicity, we only report results for MAE, but we note that we observed similar trends for NMAE as well: in particular, differences between our approaches and the baselines were found to be significant (or non-significant respectively) in the same cases, and the order of EWkNN, ECkNN and EWCkNN was the same.

5.5. Comparison of regression techniques

In this section, similarly to [38], we present our results using a default k-value of $k = 5$. In the subsequent sections, we show that the proposed approach consistently outperforms the baseline for a wide range of k-values and we discuss the effect of label noise on classification which is especially important from the point of view of potential applications.

Table 2 summarizes the results of our experiments using $k = 5$ nearest neighbors. We report mean absolute error averaged over 10 folds (in case of the Financial Tweets, Parkinsons Telemonitoring, MultiSound, Forest Fires and Wine Quality datasets) or the provided test splits (in case of the Blog Feedback dataset). For EWkNN, ECkNN and EWCkNN we provide an additional symbol $\bullet$ or $\circ$ which denotes if the performance of EWkNN, ECkNN or EWCkNN is statistically significantly better ($\bullet$) or worse ($\circ$) compared with the baseline (k-NN). The absence of $\bullet$ or $\circ$ means that the observed difference in not significant statistically.

As one can see from the comparison of kNN and EWkNN, hubness-aware weighting significantly improved the performance in all the examined cases. In contrast, error correction alone seems to have a minor effect overall: in most of the cases, there were no significant differences in terms of the performance of ECkNN and kNN. In two cases, ECkNN significantly outperformed kNN, whereas only once was ECkNN significantly worse than kNN. The combination of weighting and error correction, i.e., EWCkNN had the overall best

Table 2: Mean absolute error ± its standard deviation in case of the proposed approaches and k-NN regression. For EWkNN, ECkNN and EWCkNN we provide an additional symbol • or ◦ which denotes if the performance of EWkNN, ECkNN or EWCkNN is statistically significantly better (•) or worse (◦) compared with the baseline (k-NN). The absence of • or ◦ means that the observed difference in not significant statistically. For each dataset, the best approach is underlined.

	kNN	EWkNN	ECkNN	EWCkNN
Financial Tweets Dataset				
next day	0.0163±0.0021	0.0151±0.0021•	0.0167±0.0020	0.0148±0.0021•
next week	0.0480±0.0070	0.0444±0.0066•	0.0478±0.0061	0.0433±0.0067•
next two wks.	0.0628±0.0090	0.0581±0.0095•	0.0626±0.0081	0.0567±0.0093•
next three wks.	0.0743±0.0095	0.0687±0.0102•	0.0737±0.0097	0.0671±0.0102•
Datasets related to Parkinson's Disease				
TM total	8.2574±0.1355	8.0550±0.1673•	8.0958±0.1064•	8.0117±0.1589•
TM motor	6.4744±0.0491	6.3685±0.0879•	6.3771±0.0713•	6.3518±0.0863•
MultiSound	12.160±0.5932	10.826±0.6735•	12.340±0.4059	10.980±0.6995•
Further Datasets				
Forest Fires	19.679±7.3391	14.787±6.7381•	20.125±6.7212	14.585±6.7736•
Blog Feedback	5.6889±1.7610	5.1502±1.9076•	6.1194±1.6911◦	5.1733±1.9369•
Wine Quality	0.6163±0.0234	0.5992±0.0253•	0.6144±0.0192	0.6053±0.0219•

performance. EWCkNN was always significantly better than kNN. These results show that weighting has a higher overall impact, while error correction may further increase predictive performance.

Table 3 compares the performance of EWCkNN with further baselines MkNN, LinReg, M5P, Net-5 and Net-10. We observed that EWCkNN outperformed all of these models on the Financial Tweets, MultiSound, Forest-Fires and BlogFeedback datasets. EWCkNN outperformed MkNN on all the datasets which was expected because EWCkNN takes the badness of hubs into account while MkNN does not distinguish between good and bad hubs.

Considering the performance of EWCkNN together with the intrinsic dimensionality of the datasets, we can observe that EWCkNN performed well in cases of high intrinsic dimensionality. In particular, EWCkNN was the best out of the models compared in Table 3 whenever the intrinsic dimensionality was seven or more. This is in accordance with previous results from the literature that attribute the presence of bad hubs, and the success of hubness-aware classifier to high intrinsic dimensionality [17].

Table 3: Mean absolute error of EWCkNN and additional baselines MNkNN (with $k = 10$), LinReg, M5P, Net-5, Net-10. For MNkNN, LinReg, M5P, Net-5, Net-10 we provide an additional symbol • or ◦ which denotes if EWCkNN is statistically significantly better (•) or worse (◦) than MNkNN, LinReg, M5P, Net-5 and Net-10 respectively. The absence of • or ◦ means that the observed difference in not significant statistically. For each dataset, the best approach is underlined.

	EWCkNN	MkNN	LinReg	M5P	Net-5	Net-10
Financial Tweets Dataset						
next day	0.0148	0.1365•	0.0180•	0.0181•	0.0231•	0.0238•
next week	0.0433	0.1646•	0.0489•	0.0540•	0.0524•	0.0524•
next two wks.	0.0567	0.1823•	0.0645•	0.0659•	0.0704•	0.0708•
next three wks.	0.0671	0.1928•	0.0746•	0.0793•	0.0780•	0.0784•
Datasets related to Parkinson's Disease						
TM total	8.0117	8.3776•	8.3301•	7.2538◦	7.9401	7.7191◦
TM motor	6.3518	6.5429•	6.5384•	5.7286◦	6.1905	6.1678◦
MultiSound	10.980	12.086•	12.085•	12.169•	13.285•	14.106•
Further Datasets						
Forest Fires	14.585	19.899•	19.762•	19.938•	27.667•	28.541•
Blog Feedback	5.1733	6.3590•	8.2995•	5.9752•	5.6779•	5.8543•
Wine Quality	0.6053	0.6349•	0.5852◦	0.5523◦	0.5826◦	0.5656◦

5.6. Varying the number of nearest neighbors

We examined how k, the number of nearest neighbors influence the performance of the proposed approaches. In particular, we varied k between 1 and 10 and measured MAE of the models. This is shown in Figure 4.

As one can see, the proposed approaches are stable and have good performance for a wide range of k. More importantly, EWkNN and EWCkNN seem to converge to a better solution than kNN.

5.7. The effect of noise

On the one hand, real-world data is often affected by noise. On the other hand, appropriately selected noise models allow us to simulate the increased presence of bad hubs. Therefore we examined how the proposed methods perform in the presence of label noise. In particular, we considered two noise models, conventional Gaussian noise and the adapted version of the recently introduced hubness-aware random label noise [25]. The adaptation was necessary because hubness-aware random label noise was originally introduced for classification under the assumption of discrete class labels,

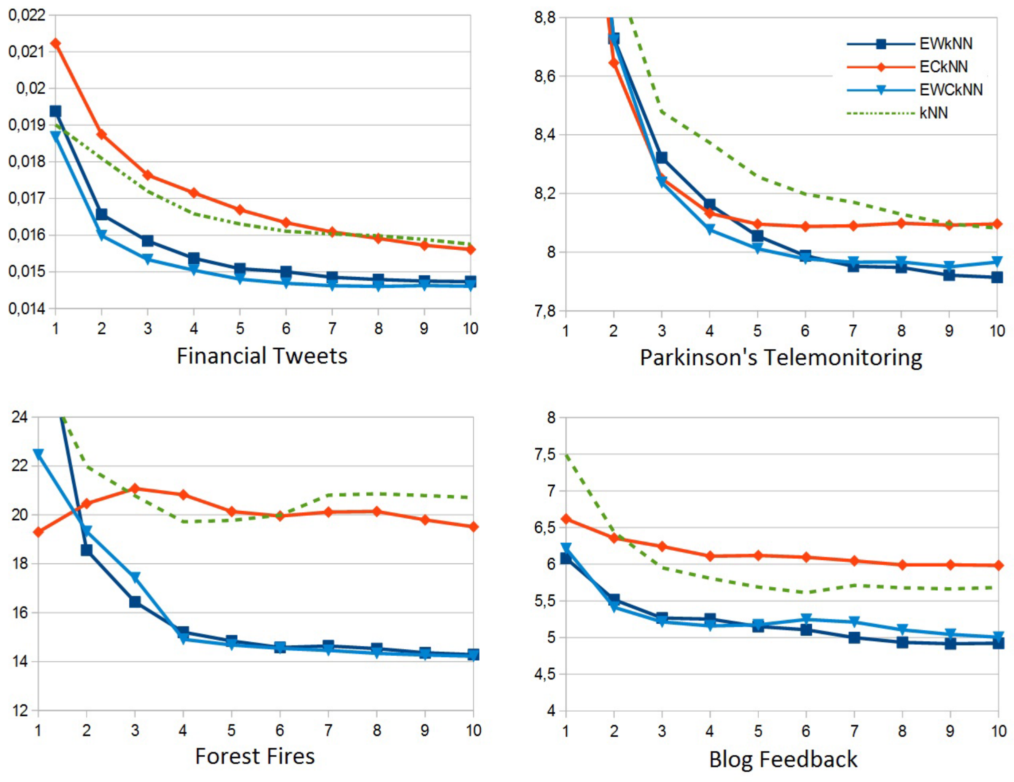


Figure 4: Performance (in MAE) of EWCkNN, EWkNN, ECkNN and kNN as function of k, the number of nearest neighbors for various datasets.

however, we consider regression problems with continuous class labels in this paper. Hubness-aware continuous random label noise introduces additional bad hubs to the data.

In case of both noise models, we added noise to the *training* data and examined how well the regression model trained on the noisy data is able to predict the correct (i.e., noise-less) class labels of the *test* data. By adding noise to the training data we mean that we changed the original labels. In case of Gaussian label noise, the noisy label $y'_G(x)$ of a training instance $x \in \mathcal{D}$ was defined as

$$y'_G(x) = y(x) + \mathcal{R} \cdot y_{avg} \cdot r \tag{14}$$

where $y(x)$ denotes the original class label of instance x, $\mathcal{R}$ is a standard normal (Gaussian) random variable with zero mean and standard deviation of one, r denotes the noise level, while y_{avg} is the average of labels of the training data, i.e.,

$$y_{avg} = \frac{1}{n} \sum_{x_i \in \mathcal{D}} y(x_i).$$

We define hubness-aware continuous random label noise as follows: in case of hubness-aware continuous random label noise, the noisy label $y'_H(x)$ of a training instance $x \in \mathcal{D}$ is

$$y'_H(x) = y(x) + \mathcal{R} \cdot N_k(x) \cdot y_{avg} \cdot r \tag{15}$$

where $N_k(x)$ is the k-occurrence of the instance x as described in Section 3, while $y(x)$, $\mathcal{R}$, y_{avg} and r denote the same as in case of Gaussian random label noise. Due to the factor $N_k(x)$ in Equation (15), hubness-aware continuous random label noise introduces bad hubs. This noise model is particularly interesting because bad hubs have been identified as one of the major causes of classification errors, see e.g. [17], [18], [19], [20], [21]. Similar noise models, although in context of classification, have been used in [25] and [39].

In our experiments presented in this section, we fixed $k = 5$ and varied the noise level r between 0 and 0.5 for both noise models. However, we note that we repeated our experiments with $k = 10$ and we observed the same trends.

The effect of Gaussian and hubness-aware continuous random label noise on the predictive performance (in MAE) of EWCkNN, EWkNN, ECkNN and kNN on the Financial Tweets and Parkinsons datasets is shown in Figure 5, Figure 6 and Figure 7. We observed similar trends on the other datasets too.

As one can see, the proposed approaches outperform the baseline for all the examined noise levels. However, there are differences in terms of the relative performance of EWkNN, ECkNN and EWCkNN. While in the noiseless case (and in case of very low levels of noise), according to our observations, the proposed weighting technique (EWkNN) contributes more to accurate predictions then the proposed label correction (ECkNN), the situation is the opposite in case of high levels of noise: with increasing noise, error correction (ECkNN) becomes more relevant then weighting (EWkNN).

In case of hubness-aware continuous random label noise, the combined approach, EWCkNN is clearly favorable. Furthermore, the results also show that, similarly to the case of Gaussian noise, in case of moderate and high noise levels, error correction (ECkNN) contributes more to accurate prediction than weighting.

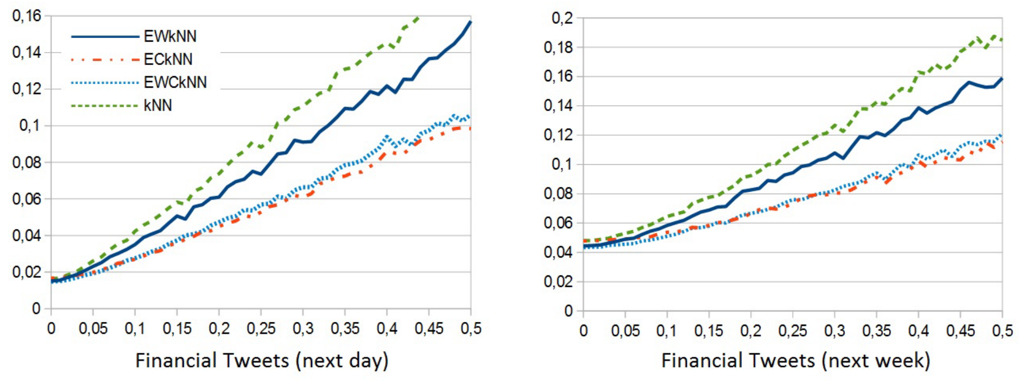


Figure 5: The effect of Gaussian noise on the predictive performance (in MAE) of EWCkNN, EWkNN, ECkNN and kNN on the Financial Tweets dataset. MAE is shown as function of the noise level.

Figure 8 shows the performance of EWCkNN compared with the performance of additional baselines LinReg, M5P, Net-5 and Net-10 under both examined noise models on the Blog Feedback dataset. As one can see, EWCkNN clearly outperforms LinReg, Net-5 and Net-10. For moderate levels of Gaussian noise, EWCkNN outperforms M5P as well, while M5P and EWCkNN have similar performance in other cases. Most remarkably, the performance of EWCkNN is much more stable than that of Net-5, Net-10 and M5P.

These results indicate that, in real-world applications, the selection of the regression approach may depend on the character and (expected) level of noise in the data.

6. Conclusions and Outlook

The presence of hubs has been previously observed and its implications to various machine learning tasks, such as classification, clustering and instance selection, has been explored. This paper is the first that describes this phenomenon in the context of regression. Here, we developed three hubness-aware regression approaches: the weighting-based EWkNN, ECkNN which performs error correction, and the combination of the two aforementioned techniques which we called EWCkNN. We evaluated these techniques on real-world datasets from various domains ranging from financial over social to medical and compared the proposed EWkNN, ECkNN and EWCkNN

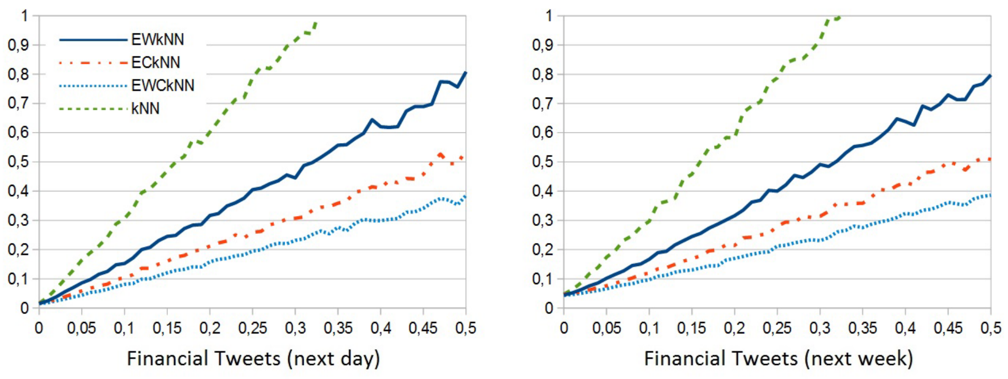


Figure 6: The effect of hubness-aware continuous random label noise on the predictive performance (in MAE) of EWCkNN, EWkNN, ECkNN and kNN on the Financial Tweets dataset. MAE is shown as function of the noise level.

to state-of-the-art regression techniques such as kNN, MkNN, regression trees and neural networks. According to our observations, EWCkNN outperformed all the aforementioned baselines in cases where the *intrinsic* dimensionality of the data was high.

We also examined the performance of the proposed approaches under the assumption of various types of noise. In particular, we considered conventional Gaussian noise model and the adapted variant of the recently proposed hubness-aware label noise which models the increased amount of bad hubs. The relevance of this noise model is due to the fact that bad hubs has been shown to be responsible for surprisingly high fraction of the overall prediction error. Our experimental results show that in case of noiseless data (i.e. data without *additional* noise), as well as in case of hubness-aware noise, the combined approach had the best overall performance.

According to our observations, out of the two components of EWCkNN, i.e., weighting and error correction, in case of noiseless data, weighting had substantially higher impact, while the presence of moderate and high levels of noise lead to increasing impact of error correction. In the presence of Gaussian noise, error correction alone (ECkNN) outperformed both the weighting-based and the combined approaches. Therefore, in real-world applications, the selection of the appropriate regression approach may depend on the character and (expected) level of noise in the data.

As regression is one of the most relevant machine learning approaches,

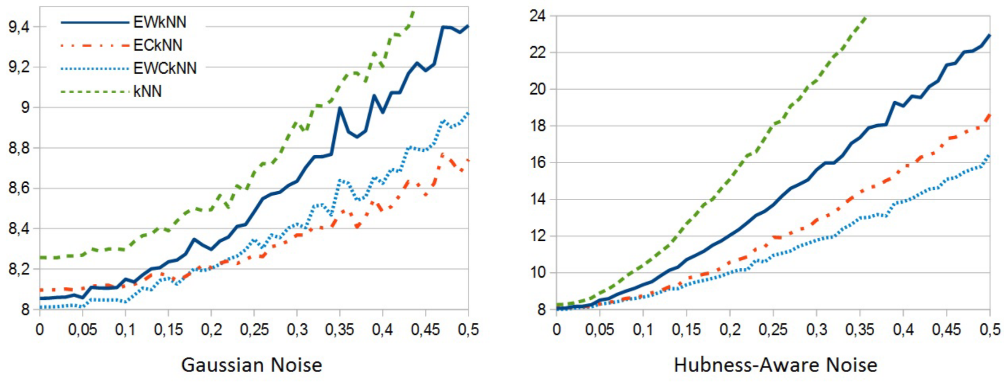
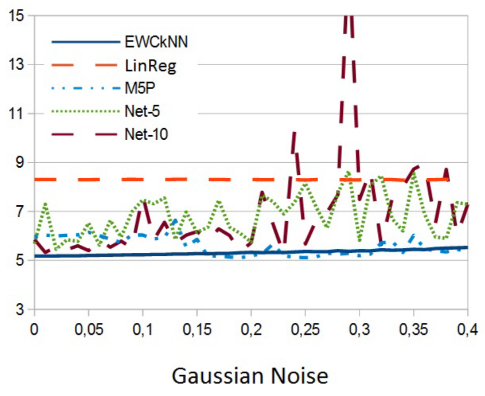


Figure 7: The effect of Gaussian and hubness-aware continuous random label noise on the predictive performance (in MAE) of EWCkNN, EWkNN, ECkNN and kNN on the Parkinsons dataset. MAE is shown as function of the noise level.

we envision that the proposed techniques can be used in various further regression tasks. Additionally, the presence of hubs may be taken into account in other ways and the proposed techniques may be combined with other conventional regression approaches in ensemble schemes. Furthermore, we point out that the proposed approach only assumes the presence of an appropriate distance between the instances and therefore it may be suited in various "big data" applications, where instances have (slightly) different attributes: in such cases, although one may define a distance between instances, most of the conventional regression techniques are likely to fail as they usually assume the presence of the same attributes for all the instances.

Acknowledgment

We thank to the Anonymous Reviewers' for their insightful comments and suggestions. Discussions about hubness-aware regression with Dr. Nenad Tomašev are highly appreciated. We thank DAAD and Magyar Ösztöndíj Bizottság for financially supporting our researcher exchange program with project number: 39859. This research was performed within the framework of the grant of the Hungarian Scientific Research Fund - OTKA 111710 PD. This paper was supported by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences. This work is partially supported by the grant: FUTURICT, TÁMOP-4.2.2.C-11/1/KONV, Financial Systems subproject.



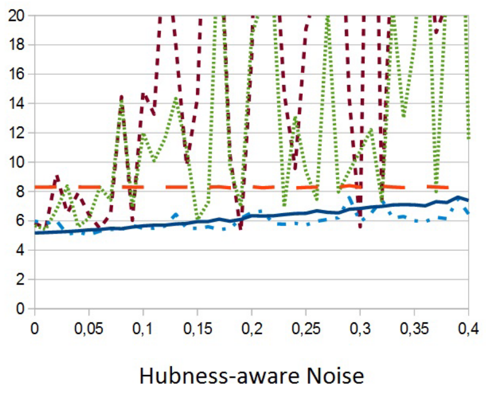


Figure 8: The effect of Gaussian and hubness-aware continuous random label noise on the predictive performance (in MAE) of EWCkNN, LinReg, M5P, Net-5 and Net-10 on the Blog Feedback dataset. MAE is shown as function of the noise level.

[1] A. Özmen, G.-W. Weber, Z. Çavuşoğlu, Ö. Defterli, The new robust conic gplm method with an application to finance: prediction of credit default, Journal of Global Optimization 56 (2) (2013) 233–249.

[2] T. Adrian, R. K. Crump, E. Moench, Efficient, regression-based estimation of dynamic asset pricing models, FRB of New York Staff Report (493).

[3] C. Hu, G. Jain, P. Zhang, C. Schmidt, P. Gomadam, T. Gorka, Data-driven method based on particle swarm optimization and k-nearest neighbor regression for estimating capacity of lithium-ion battery, Applied Energy 129 (2014) 49–55.

[4] E. B. Celikkaya, C. R. Shelton, D. Kale, R. C. Wetzel, R. G. Khemani, Non-invasive blood gas estimation for pediatric mechanical ventilation, in: Machine Learning for Clinical Data Analysis and Healthcare, NIPS Workshop, 2013.

[5] I. N. Soyiri, D. D. Reidpath, C. Sarran, Forecasting peak asthma admissions in london: an application of quantile regression models, International journal of biometeorology 57 (4) (2013) 569–578.

[6] R. Adamczak, A. Porollo, J. Meller, Accurate prediction of solvent ac-

cessibility using neural networks–based regression, Proteins: Structure, Function, and Bioinformatics 56 (4) (2004) 753–767.

[7] M. T. Leung, A.-S. Chen, H. Daouk, Forecasting exchange rates using general regression neural networks, Computers & Operations Research 27 (11) (2000) 1093–1110.

[8] D. Basak, S. Pal, D. C. Patranabis, Support vector regression, Neural Information Processing-Letters and Reviews 11 (10) (2007) 203–224.

[9] L. Devroye, L. Györfi, A. Krzyzak, G. Lugosi, On the strong universal consistency of nearest neighbor regression function estimates, The Annals of Statistics (1994) 1371–1385.

[10] G. Biau, F. Cérou, A. Guyader, On the rate of convergence of the bagged nearest neighbor estimate, The Journal of Machine Learning Research 11 (2010) 687–712.

[11] G. Biau, L. Devroye, V. Dujmović, A. Krzyżak, An affine invariant k-nearest neighbor regression estimate, Journal of Multivariate Analysis 112 (2012) 24–34.

[12] K. Stensbo-Smidt, C. Igel, A. Zirm, K. S. Pedersen, Nearest neighbour regression outperforms model-based prediction of specific star formation rate, in: Big Data, 2013 IEEE International Conference on, IEEE, 2013, pp. 141–144.

[13] H. Sakoe, S. Chiba, Dynamic programming algorithm optimization for spoken word recognition, Acoustics, Speech and Signal Processing, IEEE Transactions on 26 (1) (1978) 43–49.

[14] M. Müller, Dynamic time warping, Information retrieval for music and motion (2007) 69–84.

[15] T. F. Smith, M. S. Waterman, Identification of common molecular subsequences, Journal of molecular biology 147 (1) (1981) 195–197.

[16] V. I. Levenshtein, Binary codes capable of correcting deletions, insertions and reversals, in: Soviet physics doklady, Vol. 10, 1966, p. 707.

[17] M. Radovanović, A. Nanopoulos, M. Ivanović, Hubs in space: Popular nearest neighbors in high-dimensional data, The Journal of Machine Learning Research 11 (2010) 2487–2531.

[18] N. Tomašev, D. Mladenić, Class imbalance and the curse of minority hubs, Knowledge-Based Systems 53 (2013) 157–172.

[19] M. Radovanović, A. Nanopoulos, M. Ivanović, Nearest neighbors in high-dimensional data: The emergence and influence of hubs, in: Proceedings of the 26th Annual International Conference on Machine Learning, ACM, 2009, pp. 865–872.

[20] M. Radovanović, A. Nanopoulos, M. Ivanović, Time-series classification in many intrinsic dimensions, in: Proc. 10th SIAM Int. Conf. on Data Mining (SDM), SIAM, 2010, pp. 677–688.

[21] N. Tomašev, M. Radovanovic, D. Mladenic, M. Ivanovic, A probabilistic approach to nearest-neighbor classification: Naive hubness bayesian knn, in: Proc. CIKM, 2011.

[22] N. Tomašev, K. Buza, K. Marussy, P. B. Kis, Hubness-aware classification, instance selection and feature construction: Survey and extensions to time-series.

[23] K. Buza, A. Nanopoulos, L. Schmidt-Thieme, Insight: efficient and effective instance selection for time-series classification, in: Advances in Knowledge Discovery and Data Mining, Springer, 2011, pp. 149–160.

[24] N. Tomašev, M. Radovanović, D. Mladenić, M. Ivanović, Hubness-based clustering of high-dimensional data, in: Partitional Clustering Algorithms, Springer, 2015, pp. 353–386.

[25] N. Tomasev, K. Buza, Hubness-aware knn classification of high-dimensional data in presence of label noise, Neurocomputing.

[26] L. Györfi, A distribution-free theory of nonparametric regression, Springer Science & Business Media, 2002.

[27] A. Guyader, N. Hengartner, On the mutual nearest neighbors estimate in regression, The Journal of Machine Learning Research 14 (1) (2013) 2361–2376.

[28] N. Tomašev, K. Buza, Correcting the hub occurrence prediction bias in many dimensions, Computer Science and Information Systems.

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

[30] J. Jankovic, Parkinsons disease: clinical features and diagnosis, Journal of Neurology, Neurosurgery & Psychiatry 79 (4) (2008) 368–376.

[31] A. Tsanas, M. Little, P. McSharry, L. Ramig, Accurate telemonitoring of parkinson's disease progression by non-invasive speech tests, Biomedical Engineering, IEEE Transactions on 57.

[32] M. A. Little, P. E. McSharry, E. J. Hunter, J. Spielman, L. O. Ramig, Suitability of dysphonia measurements for telemonitoring of parkinson's disease, Biomedical Engineering, IEEE Transactions on 56 (4) (2009) 1015–1022.

[33] B. E. Sakar, M. Isenkul, C. O. Sakar, A. Sertbas, F. Gurgen, S. Delil, H. Apaydin, O. Kursun, Collection and analysis of a parkinson speech dataset with multiple types of sound recordings, Biomedical and Health Informatics, IEEE Journal of 17 (4) (2013) 828–834.

[34] P. Cortez, A. d. J. R. Morais, A data mining approach to predict forest fires using meteorological data.

[35] K. Buza, Feedback prediction for blogs, in: Data Analysis, Machine Learning and Knowledge Discovery, Springer, 2014, pp. 145–152.

[36] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, J. Reis, Modeling wine preferences by data mining from physicochemical properties, Decision Support Systems 47 (4) (2009) 547–553.

[37] I. H. Witten, E. Frank, Data Mining: Practical machine learning tools and techniques, Morgan Kaufmann, 2005.

[38] N. Tomašev, D. Mladenić, Nearest neighbor voting in high dimensional data: Learning from past occurrences, Computer Science and Information Systems/ComSIS 9 (2) (2012) 691–712.

[39] K. Buza, A. Nanopoulos, L. Schmidt-Thieme, Time-series classification based on individualised error prediction, in: Computational Science and Engineering (CSE), 2010 IEEE 13th International Conference on, IEEE, 2010, pp. 48–54.