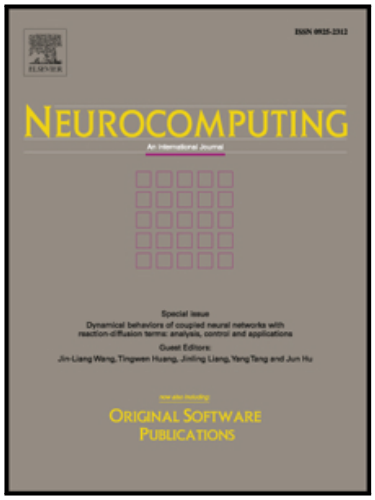
Communicated by Dr Q Wei

Accepted Manuscript



An efficient instance selection algorithm for k nearest neighbor regression

Yunsheng Song, Jiye Liang, Jing Lu, Xingwang Zhao

PII: S0925-2312(17)30688-4
DOI: 10.1016/j.neucom.2017.04.018
Reference: NEUCOM 18351

To appear in: *Neurocomputing*

Received date: 30 April 2016
Revised date: 24 January 2017
Accepted date: 3 April 2017

Please cite this article as: Yunsheng Song, Jiye Liang, Jing Lu, Xingwang Zhao, An efficient instance selection algorithm for k nearest neighbor regression, *Neurocomputing* (2017), doi: 10.1016/j.neucom.2017.04.018

An efficient instance selection algorithm for k nearest neighbor regression

Yunsheng Song[a], Jiye Liang[*,a], Jing Lu[b], Xingwang Zhao[a]

[a] *Key Laboratory of Computational Intelligence and Chinese Information Processing of Ministry of Education, School of Computer and Information Technology, Shanxi University, Taiyuan, 030006, Shanxi, China*
[b] *Shanxi Meteorological Administration, Taiyuan, 030006, Shanxi, China*

Abstract

The k-Nearest Neighbor algorithm(kNN) is an algorithm that is very simple to understand for classification or regression. It is also a lazy algorithm that does not use the training data points to do any generalization, in other words, it keeps all the training data during the testing phase. Thus, the population size becomes a major concern for kNN, since large population size may result in slow execution speed and large memory requirements. To solve this problem, many efforts have been devoted, but mainly focused on kNN classification. And now we propose an algorithm to decrease the size of the training set for kNN regression(DISKR). In this algorithm, we firstly remove the outlier instances that impact the performance of regressor, and then sorts the left instances by the difference on output among instances and their nearest neighbors. Finally, the left instances with little contribution measured by the training error are successively deleted following the rule. The proposed algorithm is compared with five state-of-the-art algorithms on 19 datasets, and experiment results show it could get the similar prediction ability but have the lowest instance storage ratio.

Key words: Instance selection, Nearest neighbor, Regression, Data reduction, Significant difference

*Corresponding author
Email addresses: sys_sd@126.com (Yunsheng Song), ljy@sxu.edu.cn (Jiye Liang), sxsqxjlws@163.com (Jing Lu), zhaoxw84@163.com (Xingwang Zhao)

1. Introduction

kNN is a kind of supervised learning method. Supervised learning infers a function(learner) from a training data T, which is a collection of training examples called samples [1]. Each sample is a pair including an input vector(instance) and the desired output value. After learning from the training set, the learner seeks to correctly determine the output for unseen instances.

In practice, the training set usually contains some noise or redundant instances, which may affect the performance of the learners on it. Thus, an increasing number of instance selection algorithms are proposed, and they aim to remove these superfluous instances from the training data. In general, the available instance selection algorithms could fall into two categories: wrapper algorithms and filter algorithms [2]. The former is a kind of algorithms that select instances based on the accuracy obtained by the learners, while the latter select the instances only relying on the training data without considering the learners. Obviously, it could be more appropriate to use wrapper algorithms for a specific learning task. The executing time of learning algorithms will be reduced after instance selection, but their generation ability could maintain relatively invariable or even be better. And instance selection algorithms for kNN are one kind of the wrapper algorithms.

Many developments have been achieved in the research on instance selection in kNN study. However, it is noteworthy that previous tests were taken on learning algorithm with the aim of classification [3], instance selection on regression remains largely understudied. Only border instances needed to be considered in classification, but not in the case of regression. Because regression is different from classification, kNN instance selection algorithm for classification cannot be used in the regression problem.

In this paper, an instance selection algorithm named DISKR is proposed to reduce training data for kNN regression. This algorithm firstly removes the outlier instances in the set T at one time and gets the set S, and then sorts the instances in the set S by defined measure. Following the sorted order, instances with less contribution to the regressor will be removed one by one; the contribution of each instance x is measured by the difference between the training error over S and the one over $S - \{x\}$. As the removed instance affects evaluating the contribution of the left instances, then it needs to reassess their contribution. However, DISKR only considers the instances whose k nearest neighbors include the removed instance according to the locality of kNN. In a word, DISKR provides a simple and effective algorithm

to distinguish which of the instances have less and negative effect to the kNN regressor. The proposed algorithm is compared with five state-of-the-art algorithms, and experimental results show that it could greatly reduce the size of training data with a similar perdition accuracy.

In total, our method is innovative. Except for reducing noise elimination, our method mainly tries to remove redundant instances to speed up executing prediction, where those instances have a little contribution to the regressor. However, previous algorithms only aim to eliminate noise information. Besides, the threshold in our method is adaptively determined; instead, it is fixed in advance for traditional threshold-based instance selection methods.

The rest of the paper is organized as follows: Section 2 briefly reviews existed instance selection algorithms. Section 3 proposes the algorithm DISKR. Section 4 presents the experiment results on the real data sets. At last, the conclusion is reached in Section 5.

2. Related work

Instance selection algorithms for regression are divided into two categories: evolutionary-based and nearest neighbor-based [4].

Tolvi [5] used a genetic algorithm that is an evolutionary based to detect the outlier in linear regression models. In this method, the corrected BIC criterion is selected as the fitness function. Each individual is fully described by a binary vector $(z_1, z_2, \cdots, z_N)$, where $z_i = 0$ indicates the instance x_i is not selected as an outlier otherwise it is selected, and $i = 1, 2, 3, \cdots, N$. The results of this experiment on small datasets have shown that it was not only able to detect the outlier, but also avoided the potential problem that one outlier prevents another one from being detected. Antonelli et al. [6] also proposed an instance selection algorithm in the framework of a multi-objective evolutionary learning of fuzzy rule-based systems. Different from the work of Tolvi, this algorithm ran on large-scale datasets and got better performance. Though the algorithms based evolutionary have better data reduction percentage and higher prediction accuracy, their computational costs are about 3 to 4 order of magnitude higher than the ones based on the neighbor for medium size data sets. Furthermore, the difference in computational cost becomes larger as the scale of data sets is growing [6]. So these methods are not easy to apply to the problems in the real life.

There also are substantial nearest neighbor-based instance selection algorithms to reduce the training data and speed up its execution [7]. One of

the research objects is the kNN. kNN algorithms are divided into two types: kNN classification and kNN regression.

According to the type of selected instances, the existing instance selection algorithms for kNN classification are classified into three categories: condensation algorithms, edition algorithms and hybrid algorithms [3]. Condensation algorithms seek to select the instances which are closer to the classification boundaries, also called border instances. The intuition behind these algorithms is that the border instances have much more effect on classification than other instances. Because there are fewer border instances in most data sets, so they could obtain a normally high reduced ratio. There are some condensation algorithms, such as CNN [8], MCS [9], POP [10], MSS [11], and so on. Different from condensation algorithms, edition algorithms always try to remove noisy instances, where those instances do not agree with their neighbors. Though the reduction ratio is low, they could improve the classification accuracy in test instances. These kinds of algorithms includes ENN [12], Multiedit [13], RNGE [14], MoCS [15], ENRBF [16], and so on. Finally, hybrid algorithms were proposed, trying to move the related instances combining the two previous strategies above. kNN classifier is highly adaptable to these algorithms. These algorithms mainly include IB3 [17], DROP3 [18], ICF [19], HMNEI [20], FCNN[21], NPPS[22] and other scalable algorithms for large-scale data [23, 24, 25]. Due to the difference between classification and regression on decision aim and error measure, the instance selection algorithms for kNN classification cannot be applied on kNN regression.

kNN regression is an important algorithm with less concern compared with kNN classification, but there are some studies on this topic. Since there are some noisy instances in data, and they take the negative effect on the performance of the regressor. Therefore, these instances should be removed first before training. For example, Guillen et al. [26] proposed a novel instance selection algorithm with mutual information in time series prediction. Although this method gets a good performance on artificially generated data, it needs to be tested on the real data sets. Fde et al. [27] present a class conditional instance selection for regression (CCISR), and it is an extension of instance selection for kNN classification. Meanwhile, CCISR has been tested on 12 real regression problems, showing a good reduction ratio while keeping the most meaningful examples. However, CCISR has higher computational time and memory demand, so that it usually exhausts the resources in practical applications[4]. In fact, a single learning algorithm is difficult

to achieve a good performance, while ensemble methods with multiple learning algorithms could obtain the better performance[28]. For this purpose, Stojanović [4] proposed the fusion of instance selection algorithms for regression tasks by ensemble idea. Compared with the original instance selection algorithms, the ensemble algorithms had the best performance on prediction error and reduced subset size. Totally, the existing methods mainly focus on noisy instances elimination, rather than the instances with less effect on the regressor. Because there are fewer noisy instances in most of the data, the reduction capability of these methods is normally low. Moreover, these methods do not consider the effect of the removed instances on the regressor, so their performance will be affected.

3. DISKR

3.1. Preliminary description

kNN regressor is based on learning by comparing the given test instances with the training set [29]. Let $T = \{(x_1, y_1), (x_2, y_2), \cdots,(x_N, y_N)\}$ be the training set with distance metrics d, where $x_i = (x_{i1}, x_{i2}, \cdots, x_{im})$ is the ith instance denoted by m attributes with its output y_i, and N is the number of instances. When given a test instance x, it needs to compute the distance d_i between x and each instance x_i in T, and sorts the distance d_i by its value. If d_i ranks in the ith place, then the distance d_i corresponding instance is called the ith nearest neighbor $NN_i(x)$, and its output is noted as $y_i(x)$. Finally the prediction output $\widehat{y}$ of x is the mean of the outputs of its k nearest neighbors in regression, i.e. , $\widehat{y} = \frac{1}{k}\sum_{i=1}^{k} y_i(x)$.

3.2. Instance selection for kNN regression

Our DISKR algorithm is mainly divided into two steps: detecting outlier instances and removing the indistinctive instances. We will introduce this algorithm in two steps in this section.

3.2.1. Detecting outliers

In statistical analysis, outlier instances are instances that are somehow different from the majority of the instances. kNN rule is easily influenced by the impact of outlier instances, which need to be removed before instance selection.

To begin, we should first remove outlier instances before the process above starts. Similarly, we remove them at one time. Based on the characters of outlier instances and their effect to the kNN regressor, the instances with the larger difference on the output with their nearest neighbors used to be outlier instances. And we define a rule to recognize the outlier instances. If $PD(x_i) = |y_i - \widehat{y}_i| > (1-\theta)y_i$, then the instance x_i is an outlier instance, otherwise it is not, where $\theta \in [0, 1]$.

3.2.2. Deleting others

Similar as instance selection for classification, the instances with more contribution to the regressor will be chosen as the representative instances. The contribution evaluation of each instance plays an important part on instance selection. However, it is difficult to directly evaluate the effect. For this reason, we propose an indirect method. If deleting an instance, the regressor would be influenced, it would be OK. Instance x_i affects the regressor as it is in T, but loses the effect as it is not in T. So the effect of x_i could be estimated by the change of performance of regressor over T and $T - \{x_i\}$. The training error is used to approximately estimate the regressor performance, and it is expressed by the residual sum of squares (RSS).

Let $R_{bf}(x_i)$ and $R_{af}(x_i)$ are the RSS over $T - \{x_i\}$ before and after the x_i is removed respectively, and we have

$$\begin{aligned} R_{bf}(x_i) &= \sum_{x_j \in T - \{x_i\}} (y_j - \widehat{y}_j)^2 \\ R_{af}(x_i) &= \sum_{x_j \in T - \{x_i\}} (y_j - \widehat{y}'_j)^2 \end{aligned} \tag{3.1}$$

where $\widehat{y}_j$ and y'_j are the predicted output of the regressor built on T and $T - \{x_i\}$, and the effect of x_i on the regressor is represented by $\nabla(x_i) = R_{af}(x_i) - R_{bf}(x_i)$.

Different from other methods, kNN is a local learning algorithm. The output of the test instance is only related to its k nearest neighbors. In order to describe better, we define a new concept **center**.

Definition 1. *If an instance x_i is one of the k nearest neighbors of x_j, then x_j is called the **Center** of x_i, noted as $Cen(x_i)$. And $\Lambda(x_i) = \{x_i \in NN(x_j) | x_j \in T\}$, is the **Influential-All-Center** of x_i,where $NN(x_j)$ is the set including all k nearest neighbors of x_j.*

$\Lambda^c(x_i)$, is the complementary set of $\Lambda(x_i)$ where $\Lambda^c(x_i) = \{x_j \in T - \{x_i\} : x_j \notin \Lambda(x_i)\}$. So $\nabla(x_i)$ can be reformulated as follows:

$$\begin{aligned}
\nabla(x_i) = & \; R_{af}(x_i) - R_{bf}(x_i) \\
= & \sum_{x_j \in T - \{x_i\}} (y_j - \widehat{y_j'})^2 - \sum_{x_j \in T - \{x_i\}} (y_j - \widehat{y_j})^2 \\
= & \{ \sum_{x_j \in \Lambda(x_i)} (y_j - \widehat{y_j'})^2 + \sum_{x_j \in \Lambda^c(x_i)} (y_j - \widehat{y_j'})^2 \} \\
- & \{ \sum_{x_j \in \Lambda(x_i)} (y_j - \widehat{y_j})^2 + \sum_{x_j \in \Lambda^c(x_i)} (y_j - \widehat{y_j})^2 \} \\
= & \sum_{x_j \in \Lambda(x_i)} \{(y_j - \widehat{y_j'})^2 - (y_j - \widehat{y_j})^2\}.
\end{aligned} \tag{3.2}$$

where $\sum_{x_j \in \Lambda^c(x_i)} (y_j - \widehat{y_j})^2 = \sum_{x_j \in \Lambda^c(x_i)} (y_j - \widehat{y_j'})^2$, because k nearest neighbors of the instances in set $\Lambda^c(x_i)$ remain the same regardless of the x_i is in or out of T. Then the computation of $\nabla(x_i)$ is simplified, and its value shows x_i has positive or negative on the regressor. If $\nabla(x_i) \leq 0$, then x_i is likely to play the positive effect on prediction; otherwise x_i is likely to take the negative effect as $\nabla(x_i) > 0$.

If an instance x_i had been deleted, each element x_j in set $\Lambda(x_i)$ has to find its $k+1$th nearest neighbor $NN_{k+1}(x_j)$ to replace x_i in its nearest neighbor list, $x_i = NN_1(x_j)$, for keeping the generality. Then the instance $NN_{k+1}(x_j)$ and the remainder nearest neighbor instances $NN_2(x_j)$, $NN_3(x_j)$, $\cdots$, $NN_k(x_j)$ reconstitute its new k nearest neighbor list, that is, $NN_2(x_j)$, $NN_3(x_j)$, $\cdots$, $NN_k(x_j)$, $NN_{k+1}(x_j)$. According to the kNN regression hypothesis, the farther distance between the instance x_j and its k nearest neighbors, the larger difference on their outputs. After this process, the difference between y_j and $\widehat{y_j'}$ could be larger than the difference between y_j and $\widehat{y_j}$, and this issue usually is true in real life. Thus $\nabla(x_i)$ may take positive value, and the performance of the regressor will have various degrees of weakness in most cases.

After an instance x_i being removed, we adopt the following rule to avoid the significant negative change on the performance of regressor, that is

$$\nabla(x_i) \leq \theta R_{bf}(x_i) \tag{3.3}$$

where $\theta \in (0, 1)$ is the significant coefficient and it is the same parameter θ in the section 3.2.1. Obviously, the larger the θ, the more the removed instances, and vice versa.

The way that the instances are removed from T plays an important role in instance selection. It is idealistic to remove all the instances meeting the removing rule from T at one time. However, this way may cause the negative change on the performance of a regressor dramatically, because a large number of instances may be removed all at once. Besides, there are some mutual center instances and all of them meet the removing rule. If we remove those instances, others would be influenced. In order to avoid the above problem, we remove the instances according to the removing rule one by one.

Algorithm 1: Decremental instance selection for kNN regression(DISKR)

Input : Dataset $T = (x_1, y_2), (x_1, y_2), \cdots, (x_N, y_N)$, the parameter θ, the number of nearest neighbor k.
Output: The subset $S \subseteq T$.

$Remove_i = 0, i = 1, 2, \cdots, N$,$S = \emptyset$;
foreach $x_i \in T$ **do**
    Computer $PD(x_i)$;
    if $PD(x_i) > (1 - \theta)y_i$ **then**
        $Remove_i = 1$
    end
end
$S = \{x_i \in T : Remove_i = 0\}$
Sort the instance x_i in S according to $PD(x_i)$ in decreasing order and obtain the set S';
foreach $x_i \in S'$ **do**
    Compute $R_{bf}(x_i)$ and $R_{af}(x_i)$;
    if $\nabla(x_i) \leq \theta R_{af}(x_i)$ **then**
        $S = S - \{x_i\}$;
        foreach $x_j \in \Lambda(x_i)$ **do**
            Find another instance x_l to replace x_j in $NN(x_j)$;
            $\Lambda(x_l) = \Lambda(x_l) \cup \{x_j\}$;
        end
    end
end
Return S

Moreover, the sequence of removing instances depends on an order, and different orders have different effects on the regressor. Therefore, we sort instances x_i by the absolute difference $PD(x_i) = |y_i - \widehat{y}_i|$ in descending order, where $\widehat{y}_i$ is the predicted output of x_i supported by its k nearest neighbors. The instance x_i with greater $PD(x_i)$ has a large divergence with its majority nearest neighbors, and there is a greater possibility that such instance has a negative impact on prediction. So these instances should be firstly tested whether they are removed or not.

Based on above content, we propose an instance selection for kNN regression, and its detail is listed in algorithm 1.

3.2.3. The determination of θ

The determination of θ value is of importance for our algorithm. Too large a value of θ results in too many instances being selected. Consequently, we will achieve a little effect of instance selection. Too small a value of θ, on the other hand, leads to a large number of instances selected. The dilemma about θ will not be symmetrical, because a serious loss of predictive accuracy of the kNN regressor is not likely to be well compensated for by a benefit of the down-sized training set.

Fixing the value of θ for every dataset is difficult, as it depends on the specific features of each problem. Therefore, we use a cross-validation approach. We divide the training set into two parts, using one of them for performing DISKR algorithm, and the other one for obtaining the validation error. Given the value of θ from small to large, the optimal θ is obtained as the last θ before the validation error starts to grow. Then, we perform the algorithm using the whole training set for the θ obtained in the cross-validation process.

4. Experimental analysis

4.1. Experiment setup

To make a fair comparison between our algorithm and others, we randomly select 19 datasets with instances larger than 1000 from the KEEL Repository[30], where this repository is a set of benchmarks to analyze the performance of the machine learning algorithms. These datasets are described in Table 1.

Table 1 Summary of 19 small datasets

Dataset	Abbreviation	Features	Size
Abalone	ABA	8	4177
Airfoil Self-Noise	AIR	6	1503
ANACALT	ANA	7	4052
California	CAL	8	20640
CASP	CAS	9	45730
CCPP	CCP	4	9568
Compaic	COM	21	8192
Concrete	CON	8	1030
Ele2	ELE	4	1056
Friedman	FRI	5	1200
House	HOU	16	22784
Mortgage	MOR	15	1049
plastic	PLA	2	1650
Pole	POL	26	14998
Quake	QUA	3	2178
Tic	TIC	85	9822
Treasury	TRE	15	1049
Wankara	WAN	9	1609
Wizmir	WIZ	9	1461

In order to compare the performance between DISKR and other instance selection algorithms, five representative algorithms for kNN regression have been selected in this study: prototype selection using mutual information(PSMI), ensemble of threshold-based CNN(TE-CNN), ensemble of threshold-based ENN (TE-ENN), ensemble of discretization-based CNN (DE-CNN) and ensemble of discretization-based ENN (DE-ENN). The selection of instance selection algorithms is based on their representativeness and popularity. Besides, CCISR is one of the typical instance selection algorithms for kNN regression, but it is out of our resources for very high computational time and memory demand for larger datasets.

The evaluation of instance selection is a key task, and they can be distinguished into two basic forms: direct and indirect evaluation. Direct evaluation aims to measure at which extent the selected instances reflect the information present in the original data. The indirect evaluation measures the performance of the learners trained on the reduced set. Generally speaking, the indirect evaluation is often used to evaluate instance selection algorithms

for instance-based learning algorithms. The coefficient of determination (R^2) and the instances compression ratio (C) are two most common metrics for the indirect evaluation. The former metric evaluates the squared correlation between the predicted and the actual value, the greater R^2, the higher prediction accuracy; the latter is the percentage of instances left, lower compression ration means a stronger reduction. 10-fold cross-validation method is used to estimate the value of these two. In this method, the data set we use is divided into 10 subsets of approximately equal size. Then the proposed method is performed 10 times, each one subset as a test set, and the remaining 9 subsets as a training set. The final result is the average of results on all the test sets.

The Wilcoxon signed rank test is used to assess the performance between DISKR and other algorithms. In this test, the null hypothesis is that there is no significant difference between DISKR and each of the other methods, against the alternative that there is a significant difference. In all of the following experiments, $k = 9$, θ from 0.05 to 1 in steps of 0.05 and the significance level $\alpha = 0.05$. The reason for choosing $k = 9$ is that the optimal k in the kNN algorithm is about 9 on most datasets due to the conclusion in[31].

4.2. DISKR behaviors

In this section, we will compare the performance of DISKR algorithm with others by the coefficient of determination R^2 and compression ratio C.

4.2.1. The coefficient of determination

The coefficient of determination R^2 is an important index to evaluate the prediction performance of the regressor, and it is independent of the data standardization. When the value of R^2 is closer to 1, the generalization ability of the regressor will be much better.

Table 2 lists the coefficients of determination R^2 of these 6 algorithms on 19 different datasets. Besides, some statistics are calculated over all data sets for each algorithm are listed in the last three rows of Table 2. Meanwhile, the trends of R^2 for these 6 algorithms on different data sets are also plotted on the Fig.1.

Table 2 R^2 of 6 algorithms on 19 different datasets

Dataset	DISKR	TE-CNN	TE-ENN	DE-CNN	DE-ENN	PSMI
ABA	0.692	0.739	0.712	0.635	0.739	0.721
AIR	0.477	0.482	0.496	0.462	0.469	0.428
ANA	0.992	0.992	0.992	0.940	0.993	0.991
CAL	0.501	0.561	0.555	0.474	0.563	0.550
CAS	0.913	0.921	0.920	0.919	0.921	0.912
CCP	0.970	0.975	0.963	0.973	0.975	0.974
COM	0.921	0.941	0.940	0.937	0.935	0.940
CON	0.797	0.855	0.762	0.862	0.732	0.845
ELE	0.997	0.996	0.996	0.997	0.996	0.997
FRI	0.940	0.953	0.943	0.944	0.952	0.942
HOU	0.307	0.314	0.364	0.315	0.301	0.300
MOR	0.998	0.996	0.959	0.996	0.983	0.991
PLA	0.859	0.874	0.845	0.861	0.879	0.855
POL	0.937	0.921	0.856	0.880	0.912	0.911
QUA	0.171	0.122	0.138	0.178	0.169	0.087
TIC	0.196	0.195	0.190	0.115	0.115	0.179
TRE	0.985	0.994	0.952	0.992	0.984	0.987
WAN	0.993	0.990	0.988	0.991	0.989	0.990
WIZ	0.996	0.994	0.996	0.996	0.996	0.996
Average	0.771	0.780	0.767	0.761	0.769	0.768
Median	0.921	0.921	0.920	0.919	0.921	0.912
Wilcixon P		0.0766	0.6292	0.7172	0.9039	0.7782

Fig.1 indicates that the values of R^2 of DISKER and another five algorithms show different change trends on these datasets. To be specific, the value of R^2 obtained by DISKER has similar or larger value than other five algorithms on the dataset ANA, CAS, CCP, ELE, FRI, PLA, POL, QUA, TIC, TRE, WAN, WIZ, and DISKER does not has the minimum value of R^2 on the remaining 7 datasets. Therefore, DISKER is not worse than these five algorithms about the index R^2 over these 19 datasets. Meanwhile, two simple statistics sample mean and median over these datasets also indicate this issue. The average value of R^2 on all the data sets of these 6 algorithms are 0.771, 0.780, 0.767, 0.761, 0.769 and 0.768, and their median values are 0.921, 0.921, 0.920, 0.919, 0.921 and 0.912. From above, the differences of R^2 among them are relatively small. In order to assess the overall performance on R^2 basing on the statistical analysis, the Wilcoxon signed rank test is used

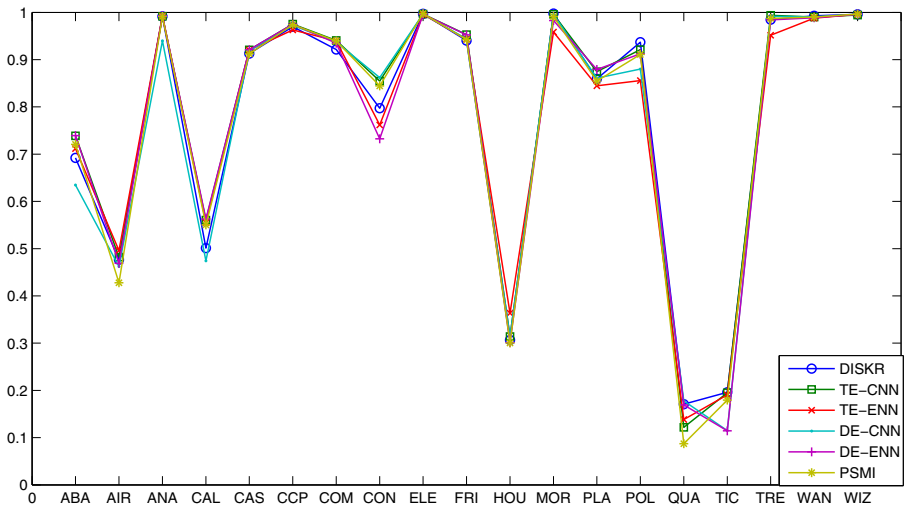


Figure 1: The R^2 of different algorithms on different datasets

to perform a paired, two-sided signed rank test between DISKER and each one of the these five algorithms. The p-value of R^2 between our DISKER and the other five algorithms are 0.0766, 0.6292, 0.7172, 0.9039 and 0.7782 respectively, and all of them are smaller than the given significant level 0.05. So it does not exist significant difference on R^2 between DISKR and each representative algorithm under given significant level 0.05.

4.2.2. The compression ration

The size of reduced subset is another key point that we mainly concern. When it comes to the storage, a low compression ratio means that it keeps less training instances, and consequently has a higher running speed. The results of compression ratio of these 6 instance selection algorithms on different data sets are shown in Table 3, and related statistic analysis is also listed in the last three rows of Table 3. At the same time, the tendency on the compression ration of different algorithms on different data sets are plotted on Fig.2.

Table 3 The storage ration of 6 algorithms on 19 different datasets

Dataset	DISKR	TE-CNN	TE-ENN	DE-CNN	DE-ENN	PSMI
ABA	0.170	0.887	0.694	0.449	0.961	0.679
AIR	0.186	0.496	0.394	0.991	0.223	0.685
ANA	0.123	0.922	0.258	0.077	0.945	0.355
CAL	0.325	0.738	0.660	0.073	0.993	0.700
CAS	0.223	0.781	0.795	0.791	0.875	0.742
CCP	0.314	0.973	0.218	0.744	0.926	0.701
COM	0.133	0.617	0.206	0.630	0.853	0.733
CON	0.452	0.757	0.714	0.999	0.412	0.828
ELE	0.549	0.705	0.889	0.216	0.992	0.834
FRI	0.457	0.836	0.847	0.595	0.995	0.624
HOU	0.137	0.425	0.470	0.365	0.385	0.623
MOR	0.735	0.997	0.818	0.753	0.859	0.729
PLA	0.204	0.281	0.158	0.532	0.807	0.812
POL	0.275	0.881	0.536	0.451	0.844	0.674
QUA	0.109	0.762	0.231	0.983	0.254	0.741
TIC	0.181	0.581	0.597	0.361	0.346	0.741
TRE	0.652	0.972	0.224	0.507	0.916	0.743
WAN	0.505	0.635	0.925	0.724	0.826	0.614
WIZ	0.462	0.999	0.900	0.719	0.908	0.683
Average	0.326	0.750	0.555	0.577	0.754	0.697
Median	0.275	0.762	0.597	0.595	0.859	0.701
Wilcixon P		1.32E-04	0.0029	0.007	1.82E-04	1.55E-04

From the Fig.2, we find that these instance selection algorithms have different storage ratio, and each algorithm has different storage ratio on different datasets. DISKR has the lowest storage ratio than other algorithms on these 11 datasets ABA, AIR, CAS, COM, FRI, HOU, POL, QUA, TIC, WAN, WIZ, and it is second-lowest storage ratio on the rest 8 datasets. According to the statistical results of Table 3, the average value of the storage ratio of these algorithms on all the data sets are 0.326, 0.750, 0.555, 0.577, 0.574 and 0.697, and their median value are 0.275, 0.762, 0.597, 0.595, 0.859 and 0.701. So DISKR has the lowest storage ratio in the term of the whole state. Furthermore, the p-value of Wilcoxon signed rank test between DISKR and each representative algorithm are 1.32E-04, 0.0029, 0.007, 1.82E-04 and 1.55E-04, they are all smaller than the given significant level 0.05. That means that

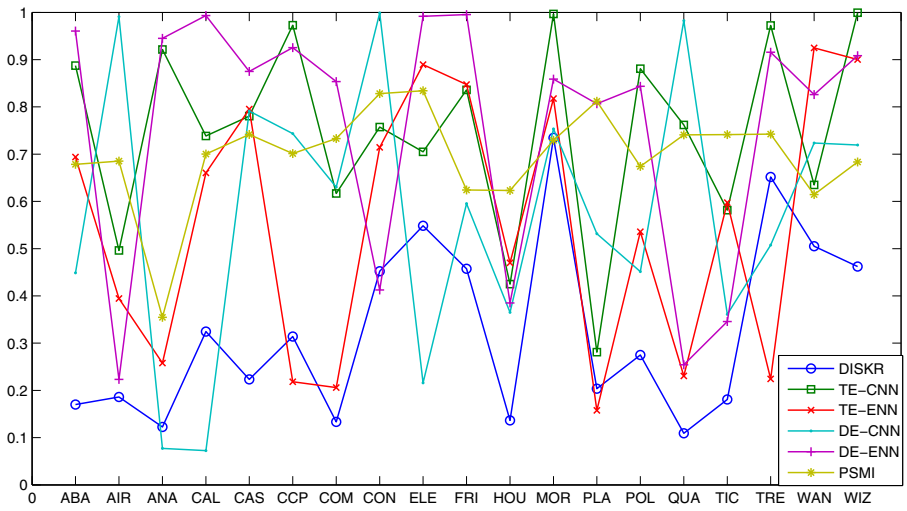


Figure 2: C obtained by different algorithms on different datasets

DISKR has the highest reduction ratio comparing with five selected algorithms. The reason for this phenomenon is that there are less number of noisy instances in most real datasets, and these five methods seek to eliminate noisy instances. While DISKR not only remove noisy instances, but the instance with less contribution to the regressor are also removed.

4.3. The effect of θ

θ controls the extent of the rule of removing instances. If θ takes smaller value, there may be many instances satisfying the rule of removing instances. Meanwhile, the less number of instances will be finally saved, and the generation ability of obtained instance subset will decrease. To this end, we change θ from 0.05 to 1 in the steps of 0.05 to perform DISKR and their results are plotted on Fig.3.

Fig. 3 presents R^2 and C obtained with DISKR over TIC dataset. The vertical axis shows R^2 and C in Fig.3 (a) and (b), and the horizontal axis shows the value of θ varying from 0.05 to 1 in the steps of 0.05. They could obviously show that the values of R^2 and C decrease as enlarging θ in whole, though their values increase as θ with small value. However, the optimal value of θ should be carefully chosen to get the tradeoff between R^2 and C.

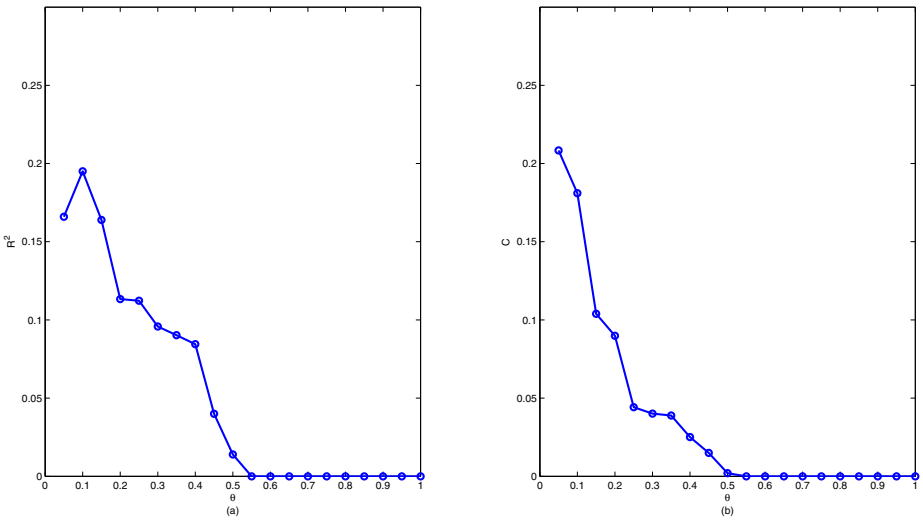


Figure 3: R^2 and C on different datasets

4.4. The effect of order

Different orders of removing instances could obtain different instance subsets, and the size and performance of these subsets are also different. We have stated that our defined order(DW) is fit for instance selection for kNN regression. However, this statement must be corroborated. In order to test the performance of DW, we choose another two different orders: random way (RW) and the way that sorts instance x_i by increasing $PD(x_i)$(IW). Their performance is shown in the Fig.4 and 5 using R^2 and C.

Compared with IW and RW, DW obtains the higher or similar value of R^2 on the most of the datasets except HOU dataset in Fig.4. Meanwhile, the p-value of Wilcoxon signed rank test between DW and each one of two other ways are 0.0158 and 0.0269, they are both smaller than the given significant level 0.05. So DW could get the better predictive ability than IW and RW. For data reduction, the value of C obtained by DW is not larger than two other ways on the most datasets except CAL, WAN, WIZ in Fig.5. And the p-value of Wilcoxon signed rank test between DW and two other ways are 0.2432 and 0.8092, they are both larger than the given significant level 0.05. Then DW gets no significant difference on data reduction with IW and RW. According to the above discussion, we can get the conclusion that DW

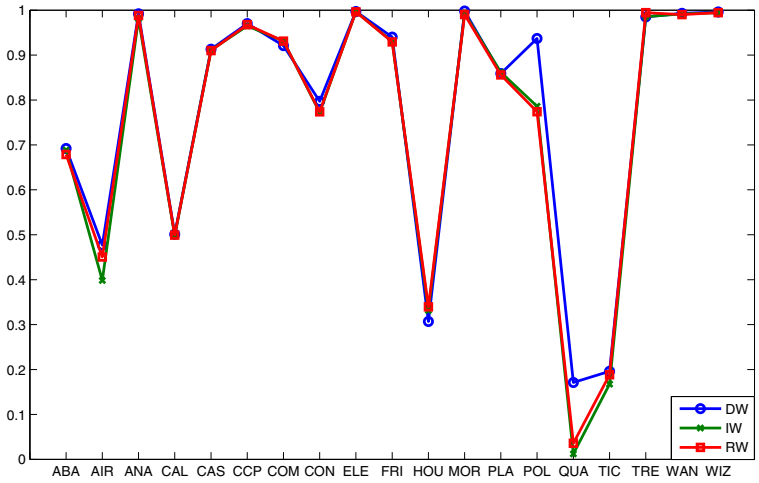


Figure 4: R^2 obtained by three different ways

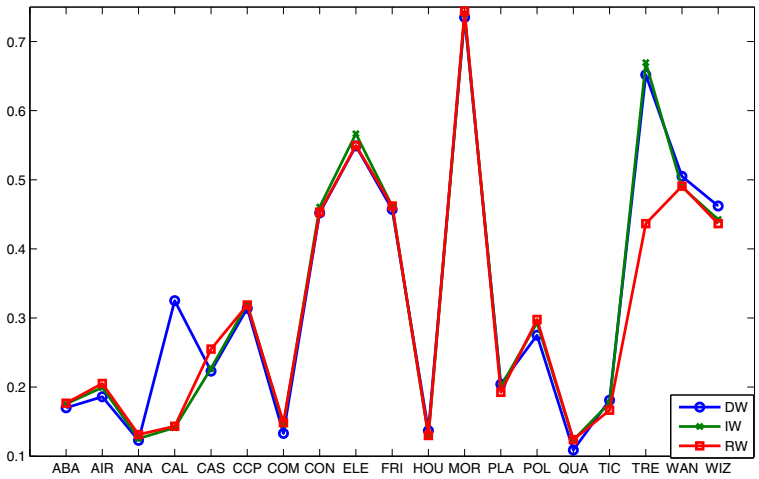


Figure 5: C obtained by three different ways

obtains the highest predictive ability with the similar data reduction ratio.

5. Conclusion

Most of the instance selection algorithms are mainly concerned with kNN classification, and less focused on kNN regression. In this paper, we propose an efficient instance selection algorithm DISKR for kNN regression. Firstly, the proposed algorithm removes the outlier instances. Secondly, it sorts the left instances by the difference between their true and predicted output provided by their neighbors. Finally, DISKER removes the instances with less effect on the regressor one by one. Experiments show that DISKR has a more consistent performance in comparison with the five state-of-the-art instance selection algorithms, but with a lower storage ratio. Thus, our method can reduce storage space, and it provides a potential method when dealing with large-scale data. In the future work, we will add the divide-and-conquer strategy to accelerate DISKER's performance for the big data.

Acknowledgement

This work is also supported by the National Natural Science Foundation of China (No.61432011, No.U1435212), the National Key Basic Research and Development Program of China (973) (No. 2013CB329404).

References

[1] C. M. Bishop, Pattern recognition and machine learning, springer, 2006.

[2] J. A. Olvera-López, J. A. Carrasco-Ochoa, J. F. Martínez-Trinidad, J. Kittler, A review of instance selection methods, Artificial Intelligence Review 34 (2) (2010) 133–143.

[3] S. Garcia, J. Derrac, J. R. Cano, F. Herrera, Prototype selection for nearest neighbor classification: Taxonomy and empirical study, IEEE Transactions on Pattern Analysis and Machine Intelligence 34 (3) (2012) 417–435.

[4] Á. Arnaiz-González, M. Blachnik, M. Kordos, C. García-Osorio, Fusion of instance selection methods in regression tasks, Information Fusion 30 (2016) 69–79.

[5] J. Tolvi, Genetic algorithms for outlier detection and variable selection in linear regression models, Soft Computing 8 (8) (2004) 527–533.

[6] M. Antonelli, P. Ducange, F. Marcelloni, Genetic training instance selection in multiobjective evolutionary fuzzy systems: A coevolutionary approach, IEEE Transactions on Fuzzy Systems 20 (2) (2012) 276–290.

[7] T. M. Cover, P. E. Hart, Nearest neighbor pattern classification, Information Theory, IEEE Transactions on 13 (1) (1967) 21–27.

[8] P. E. Hart, The condensed nearest neighbor rule, IEEE Transactions on Information Theory 14 (3) (1968) 515–516.

[9] B. V. Dasarathy, Minimal consistent set (mcs) identification for optimal nearest neighbor decision systems design, Systems, Man and Cybernetics, IEEE Transactions on 24 (3) (1994) 511–517.

[10] J. C. Riquelme, J. S. Aguilar-Ruiz, M. Toro, Finding representative patterns with ordered projections, Pattern Recognition 36 (4) (2003) 1009–1018.

[11] R. Barandela, F. J. Ferri, J. S. Sánchez, Decision boundary preserving prototype selection for nearest neighbor classification, International Journal of Pattern Recognition and Artificial Intelligence 19 (06) (2005) 787–806.

[12] D. L. Wilson, Asymptotic properties of nearest neighbor rules using edited data, Systems, Man and Cybernetics, IEEE Transactions on SMC-2 (3) (1972) 408–421.

[13] P. A. Devijver, On the editing rate of the multiedit algorithm, Pattern Recognition Letters 4 (1) (1986) 9–12.

[14] J. S. Sánchez, F. Pla, F. J. Ferri, Prototype selection for the nearest neighbour rule through proximity graphs, Pattern Recognition Letters 18 (6) (1997) 507–513.

[15] C. E. Brodley, Recursive automatic bias selection for classifier construction, Machine Learning 20 (1-2) (1995) 63–94.

[16] N. Jankowski, M. Grochowski, Comparison of instances seletion algorithms i. algorithms survey, in: Artificial Intelligence and Soft Computing-ICAISC 2004, Springer, 2004, pp. 598–603.

[17] D. W. Aha, D. Kibler, M. K. Albert, Instance-based learning algorithms, Machine learning 6 (1) (1991) 37–66.

[18] D. R. Wilson, T. R. Martinez, Reduction techniques for instance-based learning algorithms, Machine learning 38 (3) (2000) 257–286.

[19] H. Brighton, C. Mellish, On the consistency of information filters for lazy learning algorithms, in: Principles of Data Mining and Knowledge Discovery, Springer, 1999, pp. 283–288.

[20] E. Marchiori, Hit miss networks with applications to instance selection, The Journal of Machine Learning Research 9 (2008) 997–1017.

[21] F. Angiulli, Fast nearest neighbor condensation for large data sets classification, Knowledge and Data Engineering, IEEE Transactions on 19 (11) (2007) 1450–1464.

[22] H. Shin, S. Cho, Neighborhood property–based pattern selection for support vector machines, Neural Computation 19 (3) (2007) 816–855.

[23] F. Angiulli, G. Folino, Distributed nearest neighbor-based condensation of very large data sets, Knowledge and Data Engineering, IEEE Transactions on 19 (12) (2007) 1593–1606.

[24] A. de Haro-García, N. García-Pedrajas, A divide-and-conquer recursive approach for scaling up instance selection algorithms, Data Mining and Knowledge Discovery 18 (3) (2009) 392–418.

[25] C. García-Osorio, A. de Haro-García, N. García-Pedrajas, Democratic instance selection: a linear complexity instance selection algorithm based on classifier ensemble concepts, Artificial Intelligence 174 (5) (2010) 410–441.

[26] A. Guillen, L. J. Herrera, G. Rubio, H. Pomares, A. Lendasse, I. Rojas, New method for instance or prototype selection using mutual information in time series prediction, Neurocomputing 73 (10) (2010) 2030–2038.

[27] I. Rodriguez-Fdez, M. Mucientes, A. Bugarin, An instance selection algorithm for regression and its application in variance reduction, in: Fuzzy Systems (FUZZ), 2013 IEEE International Conference on, IEEE, 2013, pp. 1–8.

[28] Z.-H. Zhou, Ensemble methods: foundations and algorithms, CRC Press, 2012.

[29] T. M. Cover, Estimation by the nearest neighbor rule, Information Theory, IEEE Transactions on 14 (1) (1968) 50–55.

[30] J. Alcalá, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, F. Herrera, Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework, Journal of Multiple-Valued Logic and Soft Computing 17 (2-3) (2010) 255–287.

[31] M. Kordos, M. Blachnik, D. Strzempa, Do we need whatever more than k-nn?, in: Artificial Intelligence and Soft Computing, Springer, 2010, pp. 414–421.

Yunsheng Song is a PhD student in the School of Computer and Information Technology in Shanxi University. He received his M.S. degree from Shanxi University in 2012. His research interests are in the areas of machine learning.

Jiye Liang is a professor of School of Computer and Information Technology and Key Laboratory of Computational Intelligence and Chinese Information Processing of Ministry of Education at Shanxi University. He received his M.S. degree and Ph.D. degree from Xi'an Jiaotong University in 1990 and 2001, respectively. His current research interests include computational intelligence, granular computing, data mining and knowledge discovery. He has published more than 100 journal paper in his research fields.

Jing Lu is currently a PhD student in meteorological area at Nanjing University of Information Science and Technology. She received her Doctor degree in Computer Science from Oklahoma State University, US 2013. Her interest is machine learning, artificial intelligence and weather prediction

Xingwang Zhao is a teaching assistant in the School of Computer and Information Technology in Shanxi University. He received his M.S. degree from Shanxi University in 2011. His research interests are in the areas of data mining and machine learning.

Yunsheng Song

Jiye Liang

Jing Lu

Xingwang Zhao