Pattern Anal Applic
DOI 10.1007/s10044-015-0504-0

THEORETICAL ADVANCES

Locally adaptive *k* parameter selection for nearest neighbor classifier: one nearest cluster

Faruk Bulut[1] · **Mehmet Fatih Amasyali**[1]

Received: 9 February 2015 / Accepted: 6 July 2015

Abstract The *k* nearest neighbors (*k*-NN) classification technique has a worldly wide fame due to its simplicity, effectiveness, and robustness. As a lazy learner, *k*-NN is a versatile algorithm and is used in many fields. In this classifier, the *k* parameter is generally chosen by the user, and the optimal *k* value is found by experiments. The chosen constant *k* value is used during the whole classification phase. The same *k* value used for each test sample can decrease the overall prediction performance. The optimal *k* value for each test sample should vary from others in order to have more accurate predictions. In this study, a dynamic *k* value selection method for each instance is proposed. This improved classification method employs a simple clustering procedure. In the experiments, more accurate results are found. The reasons of success have also been understood and presented.

Keywords Dynamic *k* parameter · *k*-NN · Classification · Clustering · Meta-parameter selection

✉ Faruk Bulut
bulutfaruk@gmail.com

Mehmet Fatih Amasyali
mfatih@ce.yildiz.edu.tr

[1] Department of Computer Engineering, Faculty of Electrical & Electronics Engineering, Yıldız Technical University, Istanbul, Turkey

1 Introduction

Classifiers are divided into two main categories, *eager* and *lazy*. In contrast to *lazy* methods [e.g., *k* nearest neighbors (*k*-NN), PART, and One-Rule], *eager* ones (e.g., Decision Trees, SVM, and MLP) builds a generalized model from the training set. Basic *lazy* methods search the entire dataset for each test instance. The *k*-NN assigns the class label which is most frequent among the *k* training samples nearest to the query point. In other words, the test sample is classified into a particular class by the majority voting of the *k* closest training samples. Because of this, the closest training examples have a great influence on the classification accuracy. This memory-based classification algorithm is used with a constant *k* value defined by the user's preference. It is generally difficult to determine the best *k* value. In the literature, it is commonly recommended to assign the best *k* value for a dataset by carrying out some experiments [1]. A constant *k* value for each test instance may result low accuracy rates. This study aims to remove the side effects by proposing a novel method, dynamic *k* value selector for each test instance. To show the side effects of the constant *k* value, four evidences are presented below.

1.1 The 1st evidence

In Fig. 1, there are accuracy rates of the *k*-NN classifiers with different *k* parameters on some UCI benchmark datasets [2]. All accuracy values are obtained using 5 × 2 fold cross-validation throughout the experiments. The accuracy rates have been computed according to the constant *k* values ranging from 1 to 20 along the classification activity. When the *k* parameter changes, the accuracy level obviously changes as well. As it is seen in the *splice*

Published online: 19 July 2015

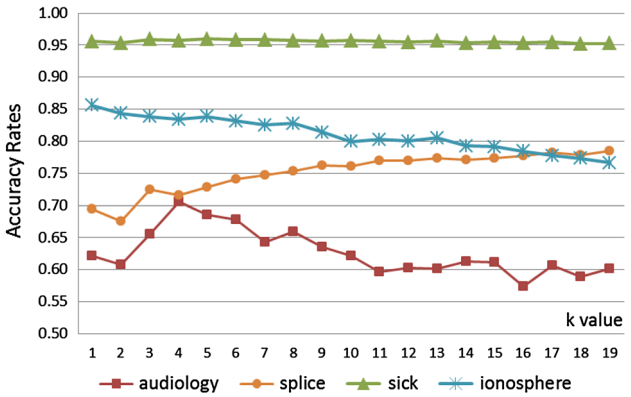

Fig. 1 Effects of the k parameter on k-NN performance

dataset, increasing the k value gives higher accuracy rate. On the other hand, in the *ionosphere* dataset, decreasing the k value gives higher accuracy rate. Besides, some datasets such as *sick* are not sensitive to the k parameter. There are only some slight changes on account of the k parameter. Nevertheless, the k value has a great influence on the *audiology* dataset. As a result, this figure shows that the k-NN method in some datasets is very sensitive to the k parameter.

1.2 The 2nd evidence

There is no unique k parameter that gives the highest cumulative accuracy rate, and Table 1 proves this. There are some randomly selected test points from the cross-validation phase. This table presents whether the different k values for randomly selected test points in the *audiology* dataset give accurate classification or not. T and F represent the true and false predictions, respectively. T indicates that the classifier predicts the test sample's class correctly and vice versa. The italic real numbers under the T and F letters indicate the probability of the prediction in percentage. An object is classified by a majority vote of its k closest neighbors. For example, the class probabilities of the 33rd test point are almost 1. This point should be in the center of a spherical group of points from the same class. The 61st test point is correctly classified when the k is bigger than 2. The estimation of the 70th test point is unstable due to different k values. It might be because the test point is near a decision boundary. This table apparently confirms that the classification performance is very sensitive to the precise value of k.

According to the table, when the k parameter is fixed to 1, only 1 class label of the test samples is truly predicted. If the k value is set to 4, 4 test points in total are accurately classified. Moreover, different k values suited for each test point increase the overall accuracy rather than constant k values. The value of k for each test point should be small, big, or in a specific interval for the purpose of higher accuracy.

1.3 The 3rd evidence

In Fig. 2, there are histogram plots of the *zoo*, *breast-cancer,* and *hepatitis* datasets from the UCI repository. In these datasets, all the values are normalized to 0–1 range. All the distances between the origin and all points have been calculated. Then, the points are laid on one-dimensional axis according to their distances to the origin to build the histogram plots. Namely, each histogram displays the frequencies of distances between the origin and all points in the space. Each histogram gives a notion about the data distribution among the origin. The histogram plots absolutely change if the test point is moved to another location of the space. Assuming a test point in the origin of the *zoo* dataset, as it is shown in the histogram plot, the best k value should be set to 3 since there are three nearest points in the same orbit. These three points are considered together due

Table 1 Analysis of the k values effecting performance

	k parameter for k-NN									
	1	2	3	4	5	6	7	8	9	10
33rd test point	T	T	T	T	T	T	T	T	T	T
	1.0	*1.0*	*1.0*	*1.0*	*1.0*	*1.0*	*1.0*	*1.0*	*0.88*	*0.9*
54th test point	F	F	F	F	F	F	F	F	F	F
	1.0	*0.5*	*0.66*	*0.75*	*0.80*	*0.83*	*0.85*	*0.75*	*0.66*	*0.60*
61st test point	F	F	T	T	T	T	T	T	T	T
	1.0	*0.5*	*1.0*	*0.75*	*0.6*	*0.83*	*0.71*	*0.75*	*0.88*	*0.9*
70th test point	F	F	F	T	T	T	F	F	F	T
	1.0	*1.0*	*1.0*	*0.75*	*0.6*	*0.66*	*0.71*	*0.62*	*0.67*	*0.6*
87th test point	F	F	F	T	F	F	F	F	F	F
	1.0	*1.0*	*0.66*	*0.75*	*0.6*	*0.66*	*0.71*	*0.75*	*0.78*	*0.7*
# of true prediction	**1**	**1**	**2**	**4**	**3**	**3**	**2**	**2**	**2**	**3**

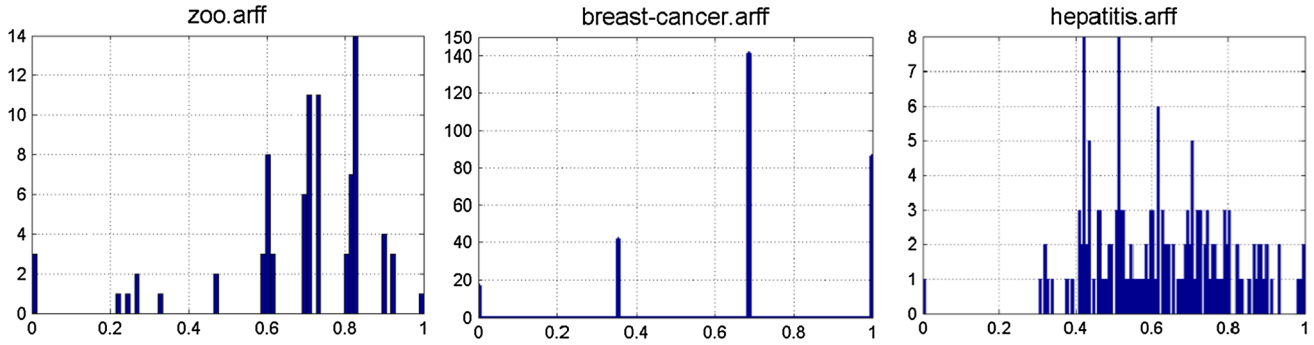

Fig. 2 Histograms of some UCI datasets

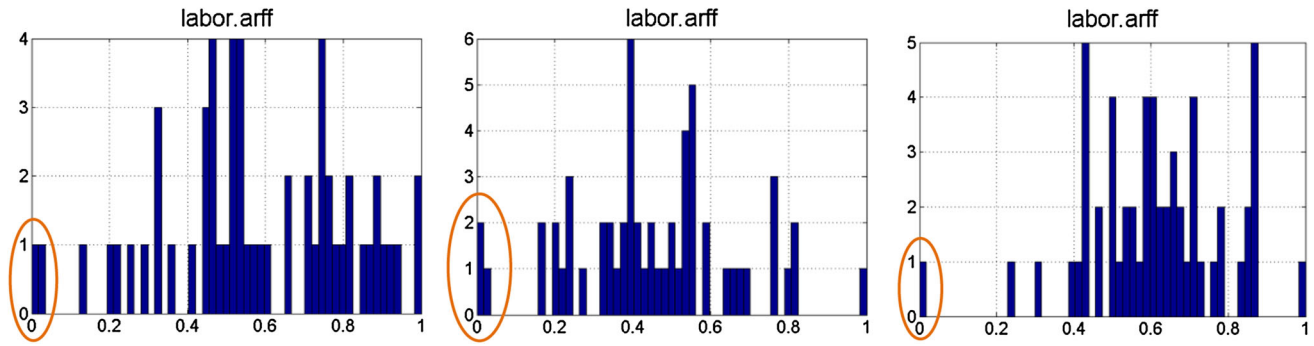

Fig. 3 Histograms of origin, center, and right upper most corner

to the same distance even though they are very far each other in their original space. Because of this, these three points are called in the same orbit. The best k value for a test point in the origin of the *breast-cancer* should be approximately 15 according to the histogram plots. In the *hepatitis* dataset, the k value should be 1. These histograms prove that there should be a suitable k value for each test point due to the distribution of its nearest neighbors on one axis.

There are three histogram plots of the same dataset *labor* in Fig. 3 showing the number of points among the origin, the center, and the upper rightmost point of the space, respectively. The histogram plots of the test instances are different from each other, although the space is the same. Instead of setting the k values for each test point to 1 in k-NN classifier, they should be 2, 3, and 1, respectively, to have reliable and accurate results. In the plots, the numbers of nearest points are circled for illustration.

1.4 The 4th evidence

There are two different scenarios in two-dimensional space illustrated in Fig. 4. The scenarios in Fig. 4a indicate that there might be other samples whose distances are similar to the test point. In this case, setting the k value to 1 results in

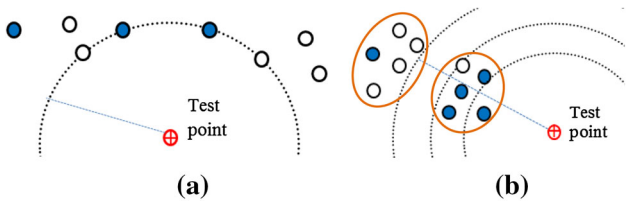

Fig. 4 Classification example

unreliable classification because of the random selection. Normally, the probability of having the same distance to the test point is very low. However, a group of samples whose distances are similar to the test point might occur as it is seen in Fig. 4b. These points might also be very close to each other. In these situations, different k values yields unstable results. Instead of this, setting the k value to the number of the samples in the nearest group produces more reliable classification.

1.5 k dependency

The performance of the classifier is strictly related to the k value. k-NN with small k values is very sensitive to the noise data. In order to reduce the effect of noise, the k value can be slightly increased. Nonetheless, if the k value exceeds its convenient level, it includes unnecessary points

from other classes and results unreliable predictions. This signifies that high k value is prone to misclassification. Additionally, attaining better accuracy with adequately high value of k requires high computational cost, because of the complexity of NN searching. As it was mentioned above, there is no exact rule and proper method of k estimation. It depends on the features of the input space such as data distribution type, structure, density of data, the number class labels, the separability of the class boundaries, and some other meta-features. In different scenarios, the optimum k might change. The value of optimum k totally depends on the dataset. In addition, small, moderate, or big k values generally yield different results. Hence, the k value may vary from dataset to dataset. The value of k is extremely training-dataset-dependent.

In addition, each section in the same dataset might be statistically different from the others in various aspects. These differences can be related with the number of samples, the number of classes, and the number of attributes. All of these features are the factors describing the sparseness and denseness of the dataset. Additionally, there might be some other geometrical meta-features [3] affecting the performance of the classifier. In order to boost the overall performance of a K-NN classifier, it will be better to build a mechanism that assigns a locally adaptive k parameter to each instance.

1.6 Drawbacks of *k*-NN

Lazy learners have expensive computational costs since they store all the training instances and do not build a generalized model. As it is known, one of the most well-known statistical pattern recognition algorithms is k-NN, and the performance of the classifier is directly related with the k parameter. Utilizing the user-defined constant k parameter during the whole learning process decreases the overall prediction performance. k-NN has two main drawbacks. The first one is the high estimation time because of the absence of a generalized model [4]. The second one is the k dependency which controls the volume of the neighborhood.

In recent years, researchers have done considerable studies on the various types of k-NN classifier such as combining it with other techniques and using different distance metrics. Jiang et al [5] presented three shortcomings of k-NN in their research. The first one is type of the distance metric for measuring the difference or similarity between two samples. The second one is artificially assigned neighborhood size as an input parameter, and finally, the third one is the class probability estimation with a simple voting technique. They propose some solutions and methods to these shortcomings and then have better performances in their experiments.

In the latest surveys of k-NN method [4] and [6], advantages and disadvantages of the popular kinds of k-NN are examined. Two main drawbacks are presented in these papers, memory requirement and computational complexity. They have analyzed several versions of this method. Three main features of these types including advantage, disadvantage, and target data are categorized in the researches. Secondly, in order to improve over memory limitations, training dataset is structured using some algorithms such as ball tree, kD-tree, nearest feature line (NFL), tunable metric, principal axis search tree, and orthogonal search tree. These methods simply decrease the computational time of NN algorithms.

We have determined that two cases where k-NN classifier fails in the experiments. These two cases are illustrated in Fig. 5. Firstly, when the k value is high, k-NN frequently fails while labeling the test samples which are closely located to the decision boundaries. Thus, the class boundaries will not be precise any more. In Fig. 5a, the vertical line represents the class boundary between white and blue classes in the two-dimensional feature space. The decision boundary is represented by the black-dotted vertical line. The red query point including the plus sign is very close to the decision boundary. It will be misclassified since the k value has been set to 5 as illustrated by the orange circle in the figure. Although the test point is in the region of blue dots, its class will be white because of the higher k value. k-NN fails here for that reason.

In Fig. 5b showing the second case, there are different neighborhood shapes where the orange ellipses and circle comprise the adaptive 8 nearest points (8NN). The shape varies with the location of the query instance due to the distribution of its nearest samples [7]. Using standard unweighted Euclidean distance metric causes misclassification in these elliptic regions. The amount of elongation/restriction decays as the query point moves further away from the elliptic regions where a decision boundary would lie. The elliptic illustrations require locally adapting a distance metric. There are some researches [7], [8] using adaptive and discrimination metrics in order to boost the performance of nearest neighbor classifiers in these two cases illustrated in Fig. 5.

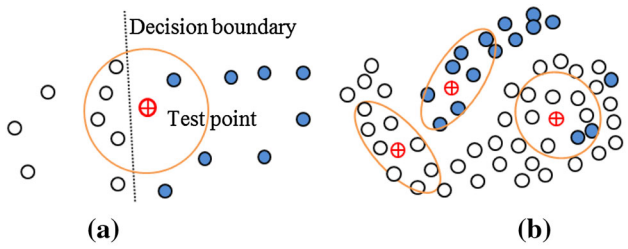

Fig. 5 Illustrations on misclassification

Patterns of different classes sometimes overlap in some regions in the feature space. Instances of different classes can be very close to the decision boundaries. In these circumstances, *k*-NN includes some irrelevant instances to the calculation. Many researchers have proposed various adaptive and discrimination metrics [8] and [7] to improve the performance. Subsequently, the adaptive *k*-NN rule with the convenient distance measure might be used to overcome these hardships for further studies.

This paper has five sections. In the first section, as mentioned above, it is proved that a new method should be built to find the best fitting *k* value for each test point in order to boost the prediction performance of the *k*-NN classifier. There are also some explanations of the reasons why *k*-NN fails on some datasets. In next section, there are briefly described studies in the related surveys and researches. In the third section, there is a definition about the dynamic *k* parameter selection procedure and search techniques used in the instance-based learning. Experimental results are presented in the fourth section. In the last section, future work is presented and our contributions are summarized.

2 Related studies

There are two main empirical evaluation approaches about setting the appropriate *k* value for a particular dataset. The *K* fold cross-validation process is the first simplest choice. Using some kind of validation process for different *k* values, the best one that gives the highest accuracy might be selected. The widely used second approach is the bootstrapping technique. It uses sampling with replacement to form the training set [9]. The optimal *k* value is determined via bootstrap method. Both approaches give approximately the same results. Although there are some suggestions [10] about setting the *k* value to the square root of the number of all training patterns, it is theoretically an upper bound value that limits these kinds of evaluations.

Ozger et al. [11] proposed an approach assigning the appropriate *k* value for a particular dataset by means of Meta-learning method. In their study, 16 meta-features are extracted from each of 200 datasets. The *k*-NN algorithms with different *k* values are computed with these datasets. In the construction of Meta-training dataset, the *k* value giving the highest accuracy becomes the class label, and the extracted meta-features are accepted as attributes. It nearly becomes possible to predict the *k* value for a specific dataset by means of this new Meta-training dataset. Therefore, it becomes possible to predict the best fitting *k* value by a regression model. However, the biggest barrier in front of the study is the assignment of the same *k* value to the whole datasets. The highest accuracy for more than half of the datasets is computed where the *k* parameter is 1. For that reason, regression becomes difficult.

In another research [12], some non-parametric *k*-NN where the general *k* for a dataset is automatically determined by geometric relationships is proposed. Classification is done by means of the centroids which globally represent the classes. Increasing the *k*-NN performance is obtained by the estimation of the optimal *k* parameter or making the *k*-NN algorithm adaptive to data by means of determining local decision boundaries.

Ghosh [13], and Guo et al. [14] have proposed some techniques that find a globally adaptive *k* value for a dataset. On the other hand, in another research [15], Ghosh has presented a locally adaptive nearest neighbor classification technique, where the value of *k* is automatically selected depending on the distribution of competing classes in the vicinity of the test point to be classified. The distribution of the nearest samples plays a great role in this technique.

Our research which differs from these studies mentioned above is about a novel way of assigning a *k* value best fitting to each test instance to gain better performance. The proposed research is aimed at contributing a dynamic *k* parameter creator to the traditional *k*-NN algorithm. The proposed hybrid method combines supervised and unsupervised techniques.

However, Euclidean is the most used distance metric in this field. It assumes that every feature of a dataset is equally important and independent from others. In literature, several distance metric learning algorithms have been proposed. Wang et al. [16] have proposed l_1-norm and Xiang et al. [17] have proposed Mahalanobis distance-based learning algorithms. In these studies, they have obtained more robust metrics to both outlier samples and irrelevant features. In Sect. 4, we have compared the proposed algorithm with the traditional NN classifiers in both Euclidean and Mahalanobis data spaces. In the experiments, the proposed algorithms have outperformed in both data spaces. Therefore, we can say that it is not relevant to the distance metric. In other and learned [16,17] data spaces, our algorithms would have better performance than the traditional NN classifiers.

3 Classification with one nearest clusters (1NC)

All the situations and scenarios mentioned in the first section indicate that there should be a different and suitable *k* parameter for each test point in the *k*-NN classification activity. The proposed hybrid algorithm which gives more accurate results is detailed in this section. The steps of the algorithm, named as *One Nearest Cluster* (1NC), are as follows:

1. Choose M, l (# of clusters), and I (# of iteration) parameters,
   Take M closest samples around the query point.
2. Calculate the distances of M closest samples to the query point,
   Normalize the distances between [0,1],
   Lay these M closest samples on one-dimensional axis by their distances to the query point
3. Split the laid samples into l groups using a basic clustering method (k-means),
4. Take all the samples in the closest cluster into the k-NN classifier and apply majority voting.

These steps are repeated for each test case. As the k parameter is generally set to small integer numbers in k-NN, it is set to 1 in the k-NC method as well.

Euclidean distance, as a most popular choice, is used to calculate the distance between two instances, even though there are some other metrics that might be preferred. The general distance formula between x and y is defined to be $\mathrm{d}(x,y)$, as follows:

$$\mathrm{d}(x,y) = \left(\sum_{i=1}^{D} |a_i(x) - a_i(y)|^r\right)^{1/r}, \tag{1}$$

where $a_i(x)$ denotes the value of the ith attribute of instance x. The arbitrary instance x in the dataset is described by the feature vector: $a_1(x), a_2(x), a(x), \ldots a_D(x)$, where D is the number of dimensions (attributes). In this formula (1), if r is set to 1, it becomes Manhattan (*Cityblock*) distance; if it is set to 2, it is Euclidean; if it is set to more than 2, it turns into *Minkowsky* distance. In the limiting case of r reaching infinity, *Chebyshev* distance is obtained. As an alternate and versatile choice, *Mahalanobis* is another metric between a point and a distribution. Apart from the others, *Mahalanobis* puts emphasis to the distributions.

The function of k-NN learner (2) predicts the class of the query point x_q as

$$\hat{f}(x_\mathrm{q}) \leftarrow \arg\max_{c \in C} \sum_{i=1}^{k} \delta(c, f(x_i)), \tag{2}$$

where $x_i \ldots x_k$ denotes the k closest training samples to x_q, and $\delta(a,\ b) = 1$ if $a = b$ and $\delta(a,b) = 0$ otherwise. The $f(x_i)$ function handles the closest k points as a target function. C is the finite set of class labels defined as $\{c_1, c_2, c_3, \ldots c_s\}$. The $f : R \rightarrow C$ function makes a calculation for each class and assigns the maximum argument to x_q as the class label.

k-NN, at the same time, can be used as a regression function as in formula (3). Here, the f function calculates the mean value of the k nearest training samples in order to label the new query point x_q.

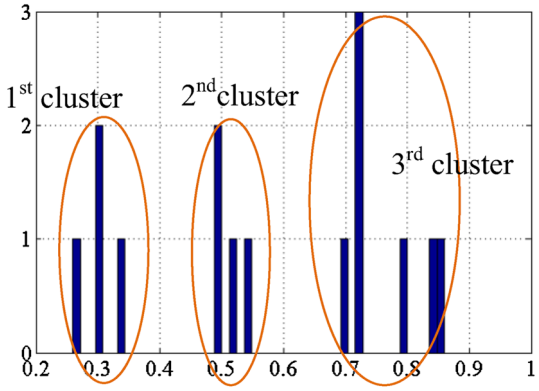

Fig. 6 The histogram of nearest points to a test point example

$$f(x_\mathrm{q}) = \frac{1}{k}\sum_{i=1}^{k} f(x_i). \tag{3}$$

Dividing the M closest points into l clusters and contributing the nearest cluster to the calculation generate a dynamic structure. It is preferred to use the l abbreviation rather than k because it is the same in both k-means and k-NN. In each of the l clusters, we can assume that there might be approximately M/l training points. The relationship between the k parameter and the M/l pair can be defined as

$$k \cong \frac{M}{l}. \tag{4}$$

The M/l combination provides a dynamic structure for each test case. The number of M/l training samples will be included in the prediction procedure. In Fig. 6, there is a real illustration about M/l pairs from the experimental steps. M/l pair is $15/3$. There is a histogram of 15 nearest samples to a test point in the *abalone* dataset. The points in the histogram are divided into three clusters by using a simple clustering method. In the 1NC technique, the closest four points in the nearest cluster will be used in the calculation of k-NN. In other words, the locally adapting k value for the current test point is set to 4.

3.1 Search methods and time complexities

As an instance- and memory-based classifier, the k-NN method searches the nearest k points in the entire dataset for each test point during test process. This operation increases the execution time. In our experiment, two types of search methods are preferred and implemented to decrease the computational time. If the number of dimension of a dataset is bigger than 10, exhaustive search method is recommended; if it is smaller than 10, kD-Tree (*k Dimensional Tree*) searcher is recommended [18], since the construction time of kD-Tree increases as the number of dimension increases. Types of these search methods do

Table 2 Comparison of the methods over 36 UCI datasets

ID	Dataset name	External features			Best k-NN results		1NN Acc.	5NN Acc.	1NC $M/l = 15/3$ Acc.	1NC-5NN comparison		1NC-1NN comparison	
		# of inst.	# of Att.	# of cls.	k	Acc.				% increase	T test	% increase	T test
1	Abalone	4153	10	19	55	0.2682	0.2023	0.2301	0.2216	−3.66	Loss	9.55	win
2	Anneal	890	62	4	1	0.9769	0.9769	0.9584	0.9699	1.20	Tie	−0.71	Tie
3	Audiology	169	69	5	4	0.7053	0.6757	0.6852	0.6367	−7.08	Loss	−5.78	Loss
4	Autos	202	71	5	1	0.6505	0.6505	0.5723	0.6188	8.13	Win	−4.87	Loss
5	Balance-scale	625	4	3	100	0.8931	0.7894	0.8576	0.8579	0.04	Tie	8.68	Win
6	Breast-cancer	286	38	2	9	0.7308	0.6643	0.7098	0.7049	−0.69	Tie	6.11	Win
7	Breast-w	699	9	2	5	0.9671	0.9548	0.9671	0.9554	−1.21	Tie	0.06	Tie
8	col10	2019	7	10	1	0.7241	0.7241	0.7072	0.7256	2.61	Win	0.10	Tie
9	Colic	368	60	2	74	0.8196	0.6957	0.7777	0.7245	−6.85	Loss	4.13	Win
10	Credit-a	690	42	2	38	0.8452	0.7901	0.8304	0.8055	−3.00	Loss	1.95	Tie
11	Credit-g	1000	59	2	14	0.7248	0.6824	0.7176	0.7022	−2.15	Tie	2.90	Win
12	d159	7182	32	2	1	0.9453	0.9453	0.9404	0.9485	0.87	Tie	0.37	Tie
13	Diabetes	768	8	2	15	0.7466	0.6943	0.7286	0.7107	−2.47	Tie	2.36	Tie
14	Glass	205	9	5	1	0.6713	0.6713	0.6410	0.6722	4.87	Win	0.13	Tie
15	Heart-statlog	270	13	2	62	0.8356	0.7467	0.8052	0.7837	−2.67	Tie	4.96	Win
16	Hepatitis	155	19	2	5	0.8361	0.7948	0.8361	0.7884	−5.71	Loss	−0.81	Tie
17	Hypothyroid	3770	31	3	5	0.9329	0.9125	0.9329	0.9306	−0.26	Tie	1.98	Tie
18	Ionosphere	351	33	2	1	0.8558	0.8558	0.8387	0.8638	2.99	Win	0.46	Tie
19	Iris	150	4	3	10	0.9693	0.9467	0.9613	0.9533	−0.83	Tie	0.70	Tie
20	kr-vs-kp	3196	39	2	3	0.9008	0.8891	0.8923	0.9293	4.14	Win	4.52	Win
21	Labor	57	26	2	1	0.8732	0.8732	0.8421	0.8611	2.25	Win	−2.35	Loss
22	Letter	20000	16	26	1	0.9441	0.9441	0.9343	0.9404	0.65	Tie	−0.36	Tie
23	Lymph	142	37	2	12	0.8338	0.7592	0.7915	0.7958	0.53	Tie	4.82	Win
24	Mushroom	8124	112	2	1	1.0000	1.0000	0.9999	0.9999	0.00	Tie	−0.01	Tie
25	Primary-tumor	302	23	11	18	0.4755	0.3874	0.4430	0.4159	−6.13	Loss	7.36	Win
26	Ringnorm	7400	20	2	2	0.7915	0.7257	0.6623	0.7361	11.15	Win	1.44	Win
27	Segment	2310	18	7	1	0.9580	0.9580	0.9443	0.9527	0.89	Tie	−0.55	Tie
28	Sick	3772	31	2	5	0.9598	0.9569	0.9598	0.9563	−0.37	Tie	−0.07	Tie
29	Sonar	208	60	2	1	0.8375	0.8375	0.7481	0.8212	9.77	Win	−1.95	Tie
30	Soybean	675	83	18	1	0.8916	0.8916	0.8776	0.8830	0.61	Tie	−0.97	Tie
31	Splice	3190	287	3	99	0.8413	0.7357	0.7285	0.7645	4.94	Win	3.91	Win
32	Vehicle	846	18	4	3	0.6839	0.6723	0.6825	0.6589	−3.46	Loss	−2.00	Tie
33	Vote	435	16	2	3	0.9297	0.9228	0.9297	0.9218	−0.84	Tie	−0.10	Tie
34	Vowel	990	11	11	1	0.9473	0.9473	0.7806	0.8972	14.93	Win	−5.29	Loss
35	Waveform	5000	40	3	75	0.8492	0.7278	0.7875	0.7519	−4.52	Loss	3.31	Win
36	Zoo	84	16	4	1	0.9976	0.9976	0.9929	0.9857	−0.72	Tie	−1.30	Tie
	Mean					0.8281	0.7943	0.7970	0.8012	3.42		2.69	

not affect the classification performance, they only has an influence on the execution time. In Exhaustive search technique, a point is found without using any algorithms and data structures. This method uses simple sequential search. The time complexity to find k closest points is $O(kDN)$, where D is the number of dimension; N is the number of points [19]. kD-Tree is one of the BSP (*Binary Space Partitioning*) methods. kD-Tree is a multi-dimensional version of BST (*Binary Search Tree*) data structure. The time complexity of finding the k closest points is

$O(kD \log N)$ due to the binary split in each dimension. The complexity of k-means notation is as $O(kNID)$ where k and I are the numbers of clusters and iterations, respectively [20].

The time complexity of the suggested method (1NC) is related to both the search and the clustering methods. Therefore, 1NC's complexity is slightly bigger than k-NN's because of the aggregation of searching, clustering, and classification complexities. The overall complexity might be written as $O\left(MD \log N + \left(\frac{M}{l}\right)ID + \left(\frac{M}{l}\right)\right)$. In this notation, the first part $(MD \log N)$ is for searching the whole dataset to find the M closest points with kD-Tree searcher; the second part $\left(\left(\frac{M}{l}\right)ID\right)$ is for clustering; and the negligible third part $\left(\frac{M}{l}\right)$ is for majority voting in k-NN. Since the number of dimension D is a constant and small integer number, it becomes an insignificant parameter. Therefore, it is sometimes not written in the notations. Finally, the notation can be summarized as $O\left(M \log N + \left(\frac{M}{l}\right)I\right)$.

4 Experimental results

Experiments are performed on 36 UCI benchmark datasets [2] which represent a wide range of data characteristics, distributions, and domains. In these real-world datasets, missing values are replaced, nominal values are converted to binary values, and finally, all the values are normalized. 5×2 fold cross-validation technique is used throughout the experiments on the MATLAB environment in order to investigate the performance of class probability estimation techniques. Although the default value of the iteration number for k-means is 100, it is decreased to 10 since the chosen nearest points in the experiments (remember $M = 15$) are very few. It has been tested that whether the iteration number 10 is enough or not. Similar results have been reached by both the iteration numbers 100 and 10. The k parameter in k-NN is set to 5 to illustrate and test our technique. All accuracy rates are computed. Paired T test metric enables to compare the overall performances of the k-NN and 1NC classifiers statistically [21]. T test gives three types of results: *win* (if the first classifier is better), *loss* (worse), and *tie* (equal). All computational results are placed in Table 2.

In Table 2, the first three columns are the number of instances, attributes, and the classes of each class, respectively. These external features describe the density and sparseness of the feature space. In the next three columns, best k-NN, 1NN, and 5NN results are presented. The next column has *One Nearest Cluster* classifier results where M/l combination is 15/3. This means the nearest 15 instances among the test point will be divided into 3 clusters. Therefore, there may be around 5 samples in each cluster. It means that the k value is set to the number of samples in the closest cluster. All samples in the nearest cluster will be computed in k-NN learning.

The accuracy comparison between 1NC $(M/l = 15/3)$ and 5NN classifiers is done by T test, and the results are seen in the "1NC-5NN comparison" column. In this comparison, there is up to %15 accuracy increment, 10 *wins*, 18 *ties*, and 8 *losses*. Additionally, 1NC and 1NN are compared in the last two columns: 12 *wins*, 20 *ties*, 4 *losses*. In the beginning, it has not been expected that 1NC would be as good as 1NN. These experimental results point out that 1NC outperforms 1NN significantly in these evaluations.

It is certain that an instance-based classifier requiring two hyper parameters is not better than k-NN including only one parameter. 1NC might be considered as a classifier requiring only one parameter by means of some experiments. These experiments handle some different M/l combinations such as 10/2, 15/3, 30/6, 50/10, 100/20, and 200/40. The results of the divisions are almost equal to 5. This means k is almost 5 in k-NN. Surprisingly, similar *win–tie–loss* results have been reached in different experiments as shown in Table 3. These similar experimental outputs of the M/l combinations provide us three consequences (inferences):

1. M/l pairs should have smaller integer numbers since different M and l values give similar outputs as seen in Table 3.
2. Small M and l values decrease the overall computational time. The computational cost of clustering remains negligible due to the small M and l values.

Table 3 The effect of different *M*/*l* pairs

T test	*M*/*l* pairs					
	10/2	15/3	30/6	50/10	100/20	200/4
# of win	7	10	8	7	8	8
# of tie	22	18	21	24	22	22
# of loss	7	8	7	5	6	6

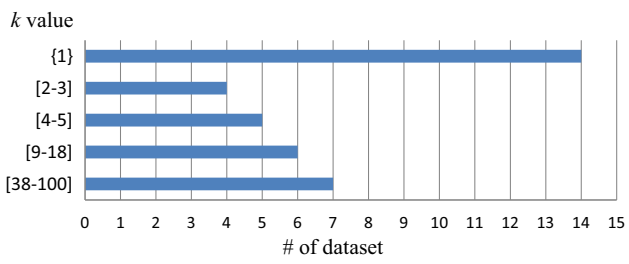

Fig. 7 Best k value of k-NN on the 36 UCI datasets

3. The M/l combination might be assumed as a unique hyper parameter of this method. If the l value is fixed, M value might be 4 or 5 times bigger than l (remember $k \cong M/l$). Consequently, a proximity between M/l and k parameters is established.

In most cases, small k parameters in k-NN give higher accuracy rates. In the experiment, 100 sorts of k-NN classifier (k value is consecutively set from 1 to 100 in ascending order) are applied to 36 UCI datasets. As shown in Fig. 7, the highest accuracy rates are computed in 14 datasets when k is equal to 1. It is evident in the figure that small k values usually provide higher accuracy rates.

The performance of the proposed method under different types of distance metric might be implemented. Since the *Mahalanobis* distance metric has a big effect on the classification results [17], it has been chosen throughout the experiments, and calculated results have been put into Table 4.

Similar successful results have been taken in this experiment. In the 1NC-5NN comparison, there are 7 wins, 9 ties, and 2 losses. Where other metrics can run over any datasets, *Mahalanobis* can run only a few datasets. Many types of metrics do not care the data distributions in a dataset. *Mahalanobis* method might produce some ill-conditioned situations on some real-world UCI datasets. Ill-conditioned covariance occurs, when the covariance matrix is reversed. Hence, experimental results can be taken from only 18 datasets.

In Table 5, the running time comparison is given. Running times in seconds have been taken by a computer having AMD A6-5200 processors, 8 GB RAM, and the MATLAB 2013a 64-bit environment. Total computational time of 1NN and 5NN over 36 datasets has been lasted for 1398.48 and 1590.30 s, respectively. But the proposed 1NC method has been lasted for 4221.98 s.

As it is seen in the table, in some big UCI datasets, the difference in the computational time between the proposed method and the k-NN method is very low. These big datasets can be defined as the bigger number of attributes and points when compared with others. On the other hand, in the small datasets, the proportional difference is higher.

5 Conclusion

In this research, a novel solution for dealing with the shortcomings of k-NN has been suggested. A mechanism is proposed with a strategy of combining lazy learning with an unsupervised learning method for two reasons: one, to eliminate the problems of the dependency on k without user's intervention; two, to increase the classification performance. The improved method using a simple clustering technique finds the appropriate k value for each of the test sample. The value of k during the classification phase varies dynamically. The well-suited dynamic k

Table 4 Comparisons of the methods using Mahalanobis distances

ID	Dataset	5NN Acc.	1NC (M/l = 15/3) Acc.	1NC-5NN comparison	
				% increase	T test
5	Balance-scale	0.8515	0.8262	−2.9665	Loss
7	Breast-w	0.9379	0.9265	−1.2191	Tie
8	col10	0.7077	0.7226	2.1104	Win
12	d159	0.9995	0.9992	−0.0335	Tie
13	Diabetes	0.7258	0.7099	−2.1912	Tie
14	Glass	0.6010	0.6390	6.3268	Win
15	Heart-statlog	0.7844	0.7674	−2.1664	Tie
16	Hepatitis	0.8065	0.8077	0.1540	Tie
18	Ionosphere	0.7521	0.8325	10.6873	Win
19	Iris	0.8560	0.8813	2.9595	Win
22	Letter	0.9214	0.9263	0.5264	Tie
26	Ringnorm	0.6466	0.7275	12.5094	Win
27	Segment	0.9087	0.9184	1.0721	Tie
29	Sonar	0.6500	0.6673	2.6628	Win
32	Vehicle	0.7589	0.7631	0.5562	Tie
33	Vote	0.9251	0.9228	−0.2531	Tie
34	Vowel	0.7216	0.8570	18.7597	Win
35	Waveform	0.5721	0.5384	−5.8976	Loss
Mean		0.7848	0.8018	4.0584	

Table 5 Computational time in seconds

Dataset ID	1NN	5NN	1NC
1	17.10	18.74	210.50
2	6.64	4.51	34.98
3	0.64	0.67	6.77
4	0.80	0.82	6.44
5	1.90	2.17	21.76
6	1.01	1.08	9.83
7	2.16	2.49	28.93
8	6.60	7.64	79.46
9	1.33	1.54	14.38
10	2.55	2.95	31.68
11	4.62	5.17	36.54
12	113.64	105.58	301.45
13	2.77	2.76	26.54
14	0.81	0.75	8.99
15	1.00	0.96	9.35
16	0.62	0.56	10.31
17	29.64	27.56	175.54
18	1.24	1.34	11.90
19	0.47	0.53	5.56
20	22.56	26.65	113.92
21	0.20	0.25	1.85
22	356.35	405.80	1007.93
23	0.48	0.54	5.35
24	459.95	519.28	764.33
25	0.97	1.40	10.83
26	71.46	108.81	342.47
27	9.12	13.83	87.42
28	25.29	40.95	165.38
29	0.74	1.02	7.83
30	3.23	4.57	25.38
31	181.51	205.90	273.33
32	2.99	3.29	38.73
33	1.42	1.60	16.93
34	3.29	3.67	38.92
35	63.07	64.61	286.87
36	0.30	0.32	3.61
Total	1398.48	1590.30	4221.98

value which is assigned to each test data adds more flexibility to the classification methodology. Experimental results carried out on 36 real-world datasets show that the proposed hybrid *k*-NN Model is a competitive method for classification. With a single parameter (M/l), more successful prediction results are reached via experiments. Nevertheless, our supervised classification method has more time complexity since it includes an unsupervised method.

6 Availability

The implemented MATLAB codes of the proposed model, the overall experimental results in Excel documents and related files can be publicly downloaded from the URL address for examinations and further studies: https://www.ce.yildiz.edu.tr/personal/mfatih/file/11271/1NC.rar.

References

1. Myatt G (2007) Making sense of data: a practical guide to exploratory data analysis and data mining , p 176–181 Wiley, New York
2. Bache K, Lichman M (2013) UCI machine learning repository. University of California, Irvine, School of Information and Computer Sciences. http://archive.ics.uci.edu/ml
3. Ho T, Basu M (2002) Complexity measures of supervised classification problems. IEEE Trans Pattern Anal Mach Intell 24(3):289–300
4. Bhatia N (2010) Survey of nearest neighbor techniques. Int J Comput Sci Inf Secur 8(2):302–305
5. Jiang L, Cai Z, Wang D, Jiang S (2007) Survey of improving K-nearest-neighbor for classification. In: Fourth international conference on fuzzy systems and knowledge discovery, vol 1, pp 679–683. doi:10.1109/FSKD.2007.552
6. Miloud-Aouidate A, Baba-Ali AR (2011) Survey of nearest neighbor condensing techniques. Int J Adv Comput Sci Appl 2(11):59–64
7. Alexandros A, Michael O, Anthony B (2013) Adaptive distance metrics for nearest neighbour classification based on genetic programming. Lecture notes in computer science, vol 7831, Springer, Heidelberg, pp 1–12
8. Wang J, Neskovic P, Cooper LN (2007) Improving nearest neighbor rule with a simple adaptive distance measure. Pattern Recognit Lett 28(2):207–213
9. Fandos R, Debes C, Zoubir AM (2013) Resampling methods for quality assessment of classifier performance and optimal number of features. Signal Process 93:2956–2968. doi:10.1016/j.sigpro.2013.05.004
10. Ghosh AK, Chaudhuri P, Murthy CA (2006) Multi-scale classification using nearest neighbor density estimates. IEEE Trans Syst Man Cybern Part B 36(5):1139–1148
11. Ozger ZB and Amasyali M (2013) KNN parameter selection via meta learning. In: Signal processing and communications applications conference (SIU), Trabzon, Turkey. doi:10.1109/SIU.2013.6531231
12. Sanchez J, Pla F, Ferri F (1997) On the use of neighbourhood-based non-parametric classifiers. Pattern Recognit Lett 18(11–13):1179–1186
13. Ghosh AK (2006) On optimum choice of k in nearest neighbor classification. Comput Stat Data Anal 50(11):3113–3123
14. Guo G, Wang H, Bell D, Bi Y, Greer K (2003) KNN model-based approach in classification. Lecture notes in computer science, vol 2888, pp 986–996. doi:10.1007/978-3-540-39964-3_62
15. Ghosh AK (2007) On nearest neighbor classification using adaptive choice of k. J Comput Gr Stat 16(2):482–502
16. Wang H, Nie F, Huang H (2014) Robust distance metric learning via simultaneous l1-norm minimization and maximization. In: Proceedings of the 31st international conference on machine, JMLR: W&CP, vol 32. Beijing, China.

17. Xiang S, Nie F, Zhang C (2008) Learning a Mahalanobis distance metric for data clustering and classification. Pattern Recognit 41:3600–3612
18. KD-tree searcher class, MathWorks. www.mathworks.com/help/stats/kdtreesearcher-class.html. Accessed 2014 Oct 22
19. Weiss MA (2013) Data structures & algorithm analysis in C++, 4th edn. Pearson, London, pp 83–85, 614–618, 629
20. Myatt G (2007) Making sense of data: a practical guide to exploratory data analysis and data mining. Wiley, New York, pp 120–129
21. Demsar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30