Dynamic k-NN classification based on subspace homogeneity

Stefanos Ougiaroglou[1], Georgios Evangelidis[2], and Konstantinos I. Diamantaras[1]

[1] Dept. of Information and Electronic Engineering,
International Hellenic University, 57400 Sindos, Thessaloniki, Greece
stoug@uom.edu.gr, kdiamant@ihu.gr
[2] Dept. of Applied Informatics, School of Information Sciences,
University of Macedonia, 54636 Thessaloniki, Greece
gevan@uom.gr

Abstract. The effectiveness of the popular k-NN classifier is highly dependent on the value of the parameter k that is chosen in advance and is fixed during classification. Different values are appropriate for different datasets and parameter tuning is usually inevitable. A dataset may include simultaneously well-separated and not well-separated classes as well as noise in certain only subspaces of the metric space. Thus, a different k value should be employed depending on the subspace where the unclassified instance lies. The paper proposes a new algorithm with five heuristics for dynamic k determination. The heuristics are based on a fast clustering pre-processing procedure that builds an auxiliary data structure. The latter provides information about the subspace where the unclassified instance lies. The heuristics exploit the information and dynamically determine how many neighbours will be examined. The data structure construction and the heuristics do not involve any input parameters. The proposed heuristics are tested on several datasets. The experimental results illustrate that in many cases they can achieve higher classification accuracy than the k-NN classifier that uses the best tuned k value.

Keywords: k-NN Classification, Dynamic k parameter determination, Homogeneous clustering, heuristics

1 Introduction

Classification is a important data mining task that has attracted the interest of the research community for many decades. Classification algorithms try to assign new, unlabeled instances to a set of predefined categories (or classes) on the basis of the available training data, namely, a set of already classified instances.

The k-NN classifier [5] predicts the class of an instance x by searching in the training set and retrieving the k nearest instances to x according to a distance metric, typically, the Euclidean distance. The nearest instances are called neighbours. Then, x is classified to the majority class among the classes that the k

nearest neighbours belong to. The majority class is determined via a procedure known as the nearest neighbours voting.

Classification accuracy highly depends on the selection of k. The value of k that achieves the highest accuracy depends on the training set used. Usually, tedious cross-validation tasks are performed to determine the "best" k value, which is then used to classify new instances. That value is unique and constant for all instances that need to be classified. Although the determination of k can not follow any general rule and the "best" k may be completely different for different training sets, large k values examine larger neighbourhoods (subspaces) and, consequently, they have to be used when the classes are not well separated and when the training set contains noise. In other words, large k values render the classifier more noise tolerant. However, large k values do not clearly define the boundaries between distinct classes. In contrast, small k values render the classifier noise sensitive and should be used on training sets with well-separated classes.

Even the "best" k value can not be optimal. Real-life datasets may have quite different form in different subspaces of the multidimensional metric space. For example, a training set may contain simultaneously well-separated and not well-separated classes as well as noise only in certain subspaces of the metric space. In such cases, a classifier that uses a fixed "best" k value may lead to less accurate classification than a classifier that utilizes a different k value for each instance that needs to be classified depending on the subspace where the latter lies. This observation triggered the motivation of the present work.

The contribution of this work is the development of a novel parameter free k-NN classifier in the sense that it uses a dynamic k value depending on nature of the subspace where the instance to be classified lies. We call the proposed classifier Subspace Homogeneity based Dynamic k-NN classifier (shd-kNN). The shd-kNN classifier utilizes heuristic methods that dynamically adjust the value of parameter k. The paper introduces five heuristics. All of them are based on a same data structure that is constructed by a simple fast and parameter-free k-means clustering [11] pre-processing procedure that builds homogeneous clusters. The data structure holds the cluster centroids as well as information about the area that each cluster centroid represents. In effect, when a new instance x needs to be classified, the nearest centroid c from the data structure is retrieved. Then based on c, k is appropriately adjusted and x is classified by searching the k nearest neighbours in the training set.

The rest of the paper is structured as follows: Section 2 briefly reviews related work and Section 3 presents in detail the proposed shd-kNN classifier and the five heuristics. The experimental study is presented in Section 4, and finally, Section 5 concludes the paper and gives directions for future work.

2 Related work

In [14] three heuristics for dynamic k value determination in the context k-NN classification are proposed. However, the three heuristics introduce parameters

that should be tuned in order to achieve high accuracy. The parameter tuning is usually computational expensive since it involves trial-and-error procedures. In [4] a clustering based method for dynamic k value selection is proposed, but involves various parameters.

Another interesting proposal to dynamically adjust the k parameter is presented in [3]. For each unclassified instance, the proposed algorithm determines the k value by constructing a hypersphere around it to capture the local distribution of the surrounding training instances. In [10], Johansson et al. propose a k-NN classifier that adopts the concept of "Spheres of Confidence" to determine the k parameter value for each instance that needs to be classified.

The work presented in [12] introduced two Adaptive kNN classifiers. They are code named Ada-kNN1 and Ada-kNN2. Ada-kNN1 uses the density and distribution of the neighborhood of each unclassified instance and learns a suitable k for it by using an artificial neural network. Ada-kNN2 uses a heuristic method guided by an indicator of the local density of the unclassified instance and information about its neighboring training instances.

Moreover, In [9] the authors demonstrated how k value can be adjusted for two-class nearest neighbour classifiers with unbalanced classes. A Bayesian method for optimum k value determination for a dataset is presented in [8]. However, the optimum k value is fixed.

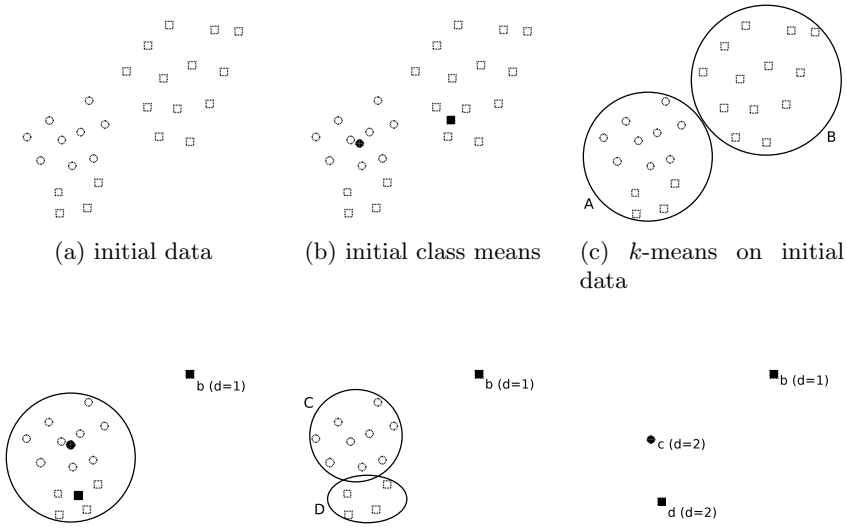
3 Subspace Homogeneity based Dynamic *k*-NN

The proposed Subspace Homogeneity based Dynamic k-NN classifier (shd-knn) is based on a simple k-means clustering procedure that builds homogeneous clusters and keeps their centroids. The result of the procedure is a data structure, which we call Structure of Homogeneous Clusters (SHC). The concept of homogeneous clustering was first presented in [13] for the purpose of developing a prototype generation data reduction technique. Here, we adopt the same methodology in order to develop a parameter free k-NN classifier that automatically adjusts k for each instance that needs to be classified depending on the nature of the subspace where the instance lies.

More specifically, SHC is build by applying the following algorithm: Initially, the training set is considered as a non-homogeneous cluster and a mean instance for each class is computed. Then, k-means clustering is applied using the class means as initial means. The result is the creation of as many clusters as the number of distinct class labels in the cluster. This clustering process is applied recursively for all non-homogeneous clusters, and in the end, all clusters become homogeneous. Each homogeneous cluster centroid is stored in SHC along with a number indicating the recursion depth, i.e., how many recursive calls were necessary to determine that homogeneous cluster.

Figure 1 presents a two dimensional example. Assume that the training set has 26 instances that can be either "squares" or "circles" (Figure 1(a)). Initially, the SHC construction algorithm computes a mean for the class "square" and a mean for the class "circle" (see Figure 1(b)). Then, k-means is executed by using

the class means as initial means and produces two clusters (Figure 1(c)). Cluster A is non-homogeneous and cluster B is homogeneous. For cluster B, the algorithm stores the cluster centroid to SHC along with the number $d = 1$, denoting that the homogeneous cluster was produced at recursion depth 1 (Figure 1(d)). For cluster A, the class means in the cluster are computed (Figure 1(d)), k-means is executed and discovers clusters C and D. Both are homogeneous (Figure 1(e)) and their centroids are placed in SHC along with the number $d = 2$ since both were produced at recursion depth 2 (Figure 1(f)).



(a) initial data (b) initial class means (c) k-means on initial data

(d) Cluster centroid and class means in a non-homogeneous cluster (e) k-means on a non-homogeneous cluster (f) final set of homogeneous cluster centroids

Fig. 1. Data generation through recursive k-means clustering

From another point of view, the aforementioned example can be illustrated as a tree of homogeneous clusters (see Figure3). The root of the tree is the whole training set. The first level of tree holds clusters A and B. Since cluster B is homogeneous, it becomes a leaf where the its centroid b is stored together with the recursion depth 1. Cluster A becomes parent of clusters C and D. Since C and D are homogeneous, they also become leaves with the corresponding centroids and their depths shown accordingly.

Obviously, for large subspaces that include instances of only one class label, the SHC construction algorithm discovers large homogeneous clusters at a relatively low recursion depth. These clusters are leaves in the cluster tree and are placed not far away from the root of the tree. In contrast, for close class border or noisy subspaces, the algorithm identifies small homogeneous clusters at higher

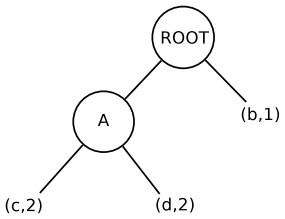


Fig. 2. Tree representation of SHC

recursion depths. Those clusters are placed far away from the root of the tree. Hence, the cluster centroids hold information about the subspace they represent in the form of the recursion depth d.

The shd-kNN classifier utilizes SHC and, based on one of the heuristics we propose below, determines the k value to be used for each individual instance that needs to be classified. Obviously, the SHC construction algorithm runs only once as a pre-processing step. Then, when a new instance x needs to be classified, shd-kNN finds the 1-nearest centroid c in SHC and its corresponding d. Then, one of the proposed heuristics is employed to determine k based on d. Finally, shd-kNN classifies x by finding the k nearest neighbours in the original training set.

The five proposed heuristics are summarized bellow:

- $k = d$: The is the simplest heuristic. It just defines k to be equal to d. For instance, if the 1-nearest centroid in SHC belongs to a homogeneous cluster that was formed at recursion depth 3, k is set to 3. One can assume that this heuristic will not be very accurate since a larger number of neighbours is often more appropriate. However, it is used as a baseline.
- $k = 2^d$: This heuristic tends to examine an extremely large number of nearest neighbours, especially when d is greater than 9. Thus, in our experiments, we manually set $k = 2^9$ when $d > 9$.
- $k = d^2$: This heuristic is a trade-off between the above two heuristics.
- $k = (d \times (d + 1))/2$ or $k = \sum_{i=1}^{d} d$: This heuristic determines k by mapping values of d to the following arithmetic sequence: $1, 3, 6, 10, 15, 21, 28, \ldots$.
- $k = \lfloor e^{\sqrt{d}} \rfloor$: This is a more conservative heuristic than the previous one. It uses the $2, 4, 5, 7, 9, 11, 14, \ldots$ sequence for determining k.

We expect that our classifier will outperform the k-NN classifiers that use fixed k parameter values for datasets that contain a mixture of well-separated and not well-separeted classes, like the well-known iris dataset. The SHC construction algorithm will build only one homogeneous cluster for the subspace containing the well-separated class (in our case iris-setosa). Thus, the proposed heuristics will consider a very small k value for each unclassified instance lying closer to that cluster centroid. In contrast, for unclassified instances close to cluster centroids in the subspaces of the other two classes the algorithm will use larger k values.

4 Performance Evaluation

4.1 Experimental Setup

The shd-kNN classifier was evaluated on fourteen datasets. Table 1 summarizes the characteristics of the datasets used. They are distributed by the KEEL dataset repository[3] [1] and by UCI machine learning repository[4] [2, 7]. We used the Euclidean distance as the distance metric. The datasets were normalized within the range $[0, 1]$.

Table 1. Dataset description

Dataset	Size	Attributes	Classes
Balance (bl)	625	4	3
Banana (bn)	5300	2	2
Ecoli (ecl)	336	7	8
Iris	120	4	3
Letter Recognition (lir)	20000	16	26
Landsat Satellite (ls)	6435	36	6
Magic G. Telescope (mgt)	19020	10	2
Pen-Digits (pd)	10992	16	10
Phoneme (ph)	5404	5	2
Pima (pm)	615	8	2
Shuttle (sh)	58000	9	7
Twonorm (tn)	7400	20	2
Texture (txr)	5500	40	11
Yeast (ys)	1484	8	10

Moreover, we wanted to test the performance of the heuristics on datasets with high levels of noise. Thus, for some of the datasets, we built two additional "noisy" versions by adding 10% and 30% random uniform noise. The noise was added by setting the class label of the 10% or 30% of the training instances to a randomly chosen different class label. The datasets on which we artificially added noise are code-named by their abbreviation plus the level of added noise (e.g., txr30). The mgt dataset has only two classes, thus, we did not build the mgt30 version for this dataset.

We compared the performance of the shd-kNN classifier against the conventional k-NN classifiers that use fixed k values. We divided each dataset into a training set and a testing set. We added noise only in the training portions. We measured the accuracy achieved by each k-NN classification approach by searching for the nearest neighbours of each testing instance in the corresponding training set. Possible ties during the majority class voting were resolved using the 1-nearest neighbour rule.

[3] `http://sci2s.ugr.es/keel/datasets.php`
[4] `http://archive.ics.uci.edu/ml/`

We build six conventional k-NN classifiers with constant k parameter value. The first three classifiers are: the widely used 1-NN classifier, the 5-NN classifier (5 is the default value for the implementation of k-NN classifier in Python's scikit-learn library [15]) and the 10-NN classifier. The fourth and fifth conventional k-NN classifiers used are those with $k = \sqrt{N}$ [6, 3] and $k = \sqrt{\frac{N}{2}}$ where N is the number of instances in the training set. They are common rule-of-thumb approaches that are often utilized in the literature. The last conventional k-NN classifier used is that with the "best" k parameter value.

"Best k" was estimated by applying a 5-fold cross validation schema. In particular, we divided each training set into five portions. Then, we ran the k-NN classifier five times. Each time, a different portion was the validation set. Each instance of the validation set was classified by applying the k-NN classifier that searches for nearest neighbours into the union of the rest four portions. The result was the average accuracy of the five executions. We applied the aforementioned procedure fifty times by varying k from 1 to 50. Then, we kept the k parameter value that achieved the highest accuracy, and this is the so called "best" k. Obviously, the accuracy achieved by the k-NN classifier that uses the "best" k is derived by classifying the instances of the initial testing set.

For the datasets with artificially added noise, we estimated the "best" k value by using validation sets without noise. To achieve this, the noise was added in a copy of the original training set. Then, the original training set and the "noisy" copy were divided into five folds. Then, we kept the five "noisy" training sets from the "noisy" copy and the five validation sets from the original training set. This procedure is described in `https://sci2s.ugr.es/noisydata`.

We conducted the experimental study without prior knowledge about the datasets. We believe that the proposed shd-kNN classifier could be more accurate than the "best" k-NN classifier only when the datasets include simultaneously well-separated and not well-separated classes as well as noise in certain subspaces. However, we conducted experiments by using datasets that may not belong to such dataset categories.

Apart from the accuracy, we estimated three computational cost measurements in terms of distance computations. The first one concerns the cost of the k-NN classifier. The k parameter value does not influence that cost. Thus, shd-kNN and conventional k-NN classifiers need to compute that number of distances. The second measurement concerns the cost overhead needed for the nearest cluster centroid searching in SHC and it concerns exclusively the shd-kNN classifiers. The last cost measurement is the pre-processing cost required for the SHC construction.

4.2 Experimental results

Table 2 presents the computational cost measurements. The last column lists the number of distances computed for SHC construction, which obviously is a computationally "cheap" algorithm. Considering that the SHC construction algorithm runs only once as a pre-processing step, the computational cost is

insignificant. The computational cost of cross-validation needed for parameter tuning in the case of "best k" is not reported. Bear in mind that it is extremely higher than the computational cost of the SHC construction algorithm.

The other two columns present the distance computations for the classification step. All the classifiers of the experimental study have to compute the distances listed in column "NN search over TS". The shd-kNN classifiers have to compute the distances that concern the search of the nearest cluster centroid in SHC. They are listed by column with header "NN search over SHC".

Table 2. Computational cost in terms of distance computations

Dataset	**NN search over TS**	**NN search over SHC**	**Construction of SHC**
bl	62,500	12,500	59,159
bl10	62,500	21,125	45,104
bl30	62,500	30,250	52,568
bn	4,494,400	940,220	592,642
ecl	18,023	6,030	41,592
ecl10	18,023	6,901	38,263
ecl30	18,023	12,596	43,200
iris	3,600	390	3,826
lir	64,000,000	7,476,000	37,168,151
ls	6,625,476	679,536	1,751,252
ls10	6,625,476	1,537,965	1,945,468
ls30	6,625,476	2,626,767	1,968,248
mgt	57,881,664	12,035,856	3,830,966
mgt10	57,881,664	17,296,788	3,918,354
pd	19,329,212	655,004	2,593,601
pd10	19,329,212	5,303,774	4,231,769
pd30	19,329,212	10,132,780	4,298,161
ph	4,669,920	886,680	639,564
ph10	4,669,920	1,710,720	693,068
ph30	4,669,920	2,494,800	766,212
pm	94,095	27,846	59,688
pm10	94,095	35,343	58,802
pm30	94,095	36,261	65,418
sh	538,193,600	1,774,647	10,977,178
tn	8,761,600	307,840	1,564,256
tn10	8,761,600	1,147,000	1,590,280
tn30	8,761,600	1,863,320	1,793,078
txr	4,840,000	257,400	2,623,438
txr10	4,840,000	1,250,700	3,403,100
txr30	4,840,000	2,523,400	3,521,357
ys	351,648	176,712	431,125
ys10	351,648	204,832	581,014
ys30	351,648	256,336	390,432

Tables 3, 4 and 5 present the accuracy measurements. Almost in all cases an shd-kNN classifier can achieve higher accuracy than the accuracy achieved by the conventional k-NN classifiers with $k = 1$, $k = 5$, $k = 10$, $k = \sqrt{N}$ and $k = \sqrt{\frac{N}{2}}$. It is worth mentioning that 10-NN performs quite well on the specific suite of datasets, since a large number of them contain noise. It turns out that 10-NN is on average better than 1-NN or 5-NN.

Notice, though, that shd-kNN classifiers in many cases clearly beat all versions of kNN with fixed k (excluding *best* k-NN). This is demonstrated in the bl, bl10, ecl30, iris, ph30, pm10, txr30 and ys datasets. Finally, shd-kNN classifiers almost always outperform RoT classifiers. This is the reason that in tables 3 and 4 we indicate with boldface only the winners among *best* k-NN and the shd-kNN classfiers.

The comparison between the *best* k-NN and shd-kNN classifiers reveals noteworthy performance for shd-kNN. At least one of the shd-kNN classifiers can achieve higher accuracy than that of the *best* k-NN classifier in 18 datasets, while in two datasets, a shd-kNN approach is as accurate as the *best* k-NN classifier. It is worth mentioning that contrary to the *best* k-NN classifier, shd-kNN achieves that performance without the need of any input parameter and tedious and costly parameter tuning procedures.

The $k = (d \times (d + 1))/2$ heuristic seems to be an ideal approach since it achieves high accuracy even when the dataset contains noise. In nine datasets it is more accurate than *best* k-NN classifier. The simple $k = d$ heuristic performs well on datasets that in their original form do not include noise (e.g., lir, pd, sh, txr). The $k = 2^d$, $k = d^2$ and $k = \lfloor e^{\sqrt{d}} \rfloor$ heuristics all achieve accuracy measurements that are close to those of the *best* k-NN classifier and, in some cases, even better.

The comparison between shd-kNN and *best* k-NN classifiers on "noisy" versions of the datasets does not reveal any useful insights on which classifier is more accurate on "noisy" conditions. The noise in the data leads to smaller clusters with high d values. Thus, all shd-kNN heuristics use a high k. Accordingly, the tuning process through cross-validation reveals a high k value for the *best* k-NN classifier.

Table 6 gives an interesting insight on the determination of k based on d. For each dataset, we report the number of testing instances whose 1-nearest cluster centroid in SHC is at a given recursion depth. We observe that in general the number of instances follow the normal distribution for almost all datasets. There are some interesting exceptions, like the ph dataset, where a large number of instances are assigned to a small depth. Also, in many datasets the distribution is left-skewed. As expected, we observe shifted to the right distributions (larger d values) for noisy datasets.

Table 3. k-NN Classification with fixed and dynamic k parameter values (datasets bl-ph30)

Dataset	**"Best" k value**	**"Best" k-NN**	**1-NN**	**5-NN**	**10-NN**	**RoT** $k = \sqrt{N}$	**RoT** $k = \sqrt{\frac{N}{2}}$	**shd-kNN** $k = d$	$k = 2^d$	$k = d^2$	$k = (d \times (d+1))/2$	$k = \lfloor e^{\sqrt{d}} \rfloor$
bl	42	89.60	79.20	86.40	89.60	89.60	89.60	84.80	88.80	89.60	**90.40**	89.60
bl10	43	88.80	68.80	86.40	87.20	89.60	89.60	88.00	**91.20**	89.60	90.40	87.20
bl30	32	**88.00**	59.20	69.60	79.20	83.20	83.20	73.60	83.20	87.20	82.40	76.80
bn	29	**90.66**	87.26	89.81	90.19	90.57	90.47	89.81	88.68	90.38	90.28	90.47
ecl	6	**91.05**	83.58	89.55	92.54	86.57	92.54	88.06	88.06	86.57	86.57	89.55
ecl10	6	89.55	77.61	88.06	92.54	88.06	89.55	86.57	91.05	**92.54**	91.05	91.05
ecl30	14	85.08	61.19	82.09	85.08	85.08	85.08	76.12	85.08	**88.06**	**88.06**	85.08
iris	11	93.33	90.00	93.33	93.33	93.33	93.33	93.33	93.33	93.33	**96.67**	93.33
lir	4	**95.78**	95.70	95.40	95.03	81.05	84.20	95.65	95.28	93.83	95.05	95.43
ls	8	**91.53**	89.98	91.53	91.14	86.09	87.34	91.22	90.68	89.59	90.99	90.83
ls10	8	**91.30**	81.66	90.52	90.99	86.09	87.10	90.75	90.37	89.90	89.98	90.75
ls30	13	88.27	61.23	82.21	89.04	86.25	86.79	82.98	88.58	89.36	**88.81**	87.72
mgt	10	**83.57**	80.13	82.97	83.57	80.97	81.65	83.36	78.97	80.86	81.86	83.39
mgt10	20	**83.10**	73.32	80.97	82.76	81.13	81.55	82.89	78.68	80.81	81.84	82.99
pd	1	99.05	99.05	99.09	98.73	95.04	96.13	**99.09**	98.59	98.32	98.54	98.91
pd10	8	98.86	89.40	98.95	98.73	95.13	96.36	**98.91**	98.73	98.36	98.64	**98.91**
pd30	12	**98.68**	69.75	94.18	98.45	94.90	96.36	92.58	98.04	98.23	98.41	97.59
ph	1	**88.70**	88.70	86.94	86.30	82.04	83.52	85.74	78.98	81.30	81.94	84.44
ph10	8	**86.67**	81.39	86.02	86.02	81.76	83.89	84.35	78.80	81.11	81.48	84.26
ph30	48	80.19	65.46	72.32	75.28	80.93	80.65	77.96	79.07	80.28	**80.93**	79.44

Table 4. k-NN Classification with fixed and dynamic k parameter values (datasets pm-ys30)

Dataset	**"Best"** k **value**	**"Best"** k**-NN**	**1-NN**	**5-NN**	**10-NN**	**RoT** $k = \sqrt{N}$	**RoT** $k = \sqrt{\frac{N}{2}}$	**shd-kNN** $k = d$	$k = 2^d$	$k = d^2$	$k = (d \times (d+1))/2$	$k = \lfloor e^{\sqrt{d}} \rfloor$
pm	48	76.47	77.12	78.43	81.05	75.16	77.78	80.39	72.55	74.51	73.86	**82.35**
pm10	18	78.43	72.55	76.47	78.43	75.16	78.43	75.82	77.12	77.78	**79.09**	77.78
pm30	47	73.86	64.05	64.71	69.94	71.90	71.24	71.24	73.86	71.90	73.20	**76.47**
sh	1	**99.96**	99.96	99.91	99.86	99.44	99.36	99.89	99.72	99.67	99.83	99.85
tn	49	97.97	95.54	97.70	97.77	97.77	98.04	97.77	**98.11**	97.70	97.91	98.04
tn10	47	97.77	84.39	95.47	97.10	97.57	97.50	97.10	**97.91**	97.43	97.70	97.70
tn30	50	**97.77**	67.23	79.19	86.08	97.43	97.57	86.69	**97.77**	97.03	97.57	94.73
txr	1	**98.73**	98.73	98.09	97.82	94.73	95.82	98.36	98.09	97.55	98.00	98.09
txr10	5	97.91	90.55	97.91	97.82	94.55	95.55	**98.09**	97.73	97.46	97.64	97.73
txr30	10	**97.55**	69.73	93.55	97.55	94.82	94.91	91.09	96.55	97.00	**97.55**	96.64
ys	14	58.45	49.32	57.10	59.12	61.15	60.14	55.41	61.15	60.14	**61.49**	57.43
ys10	13	58.45	44.93	52.37	58.11	60.14	58.11	51.01	59.46	**59.80**	57.43	57.43
ys30	14	56.76	36.82	45.61	55.41	58.78	59.46	44.26	50.00	**57.77**	56.08	48.99

Table 5. k-NN Classification with fixed and dynamic k parameter values (Average measurements)

Dataset	**"Best"** k **value**	**"Best"** k**-NN**	**1-NN**	**5-NN**	**10-NN**	**RoT** $k = \sqrt{N}$	**RoT** $k = \sqrt{\frac{N}{2}}$	**shd-kNN** $k = d$	$k = 2^d$	$k = d^2$	$k = (d \times (d+1))/2$	$k = \lfloor e^{\sqrt{d}} \rfloor$
AVG:	19.42	**87.63**	76.77	84.33	86.72	85.33	86.15	84.63	86.19	86.82	87.02	86.70

5 Conclusions

We propose the shd-kNN classifier, a parameter free k-NN classifier that heuristically determines how many neighbours will be examined for each under classification instance. Hence, a different k value is employed depending on the subspace where the instance lies. The classifier uses a pre-processing step that builds an auxiliary data structure (SHC) corresponding to the centroids of homogeneous clusters obtained via a k-Means clustering procedure together with their depth (a number indicating the recursion depth when the clusters were formed). When an instance needs to be classified, SHC provides information about the subspace where the instance lies, in effect, whether the instance is in a noisy subspace in terms of class labels or not. Then the heuristic used exploits the information and dynamically determines how many neighbours will be examined.

The data structure construction and the heuristics do not involve input parameters. The proposed heuristics are tested on several datasets. The experimental results illustrate that in many cases they can achieve higher classification accuracy than the k-NN classifier that uses the best tuned k value.

We plan to further explore dynamic k parameter determination in the context of the k-NN classifier. More specifically, we plan to develop heuristics that take into consideration not only the depth of the cluster centroid but additional metrics about the subspace such as the number of instances the centroid represents and the number of distinct class labels that exist in the immediate neighborhood of the instance.

References

1. Alcalá-Fdez, J., Fernández, A., Luengo, J., Derrac, J., García, S.: KEEL data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. Multiple-Valued Logic and Soft Computing **17**(2-3), 255–287 (2011)
2. Bache, K., Lichman, M.: UCI machine learning repository (2013), `http://archive.ics.uci.edu/ml`
3. Bhattacharya, G., Ghosh, K., Chowdhury, A.S.: Test point specific k estimation for knn classifier. In: Proceedings of the 2014 22nd International Conference on Pattern Recognition. p. 14781483. ICPR 14, IEEE Computer Society, USA (2014). https://doi.org/10.1109/ICPR.2014.263, `https://doi.org/10.1109/ICPR.2014.263`
4. Bulut, F., Amasyali, M.F.: Locally adaptive k parameter selection for nearest neighbor classifier: One nearest cluster. Pattern Anal. Appl. **20**(2), 415425 (May 2017). https://doi.org/10.1007/s10044-015-0504-0, `https://doi.org/10.1007/s10044-015-0504-0`
5. Dasarathy, B.V.: Nearest neighbor (NN) norms : NN pattern classification techniques. IEEE Computer Society Press (1991)
6. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification (2nd Edition). Wiley-Interscience, USA (2000)
7. Frank, A., Asuncion, A.: UCI machine learning repository (2010), `http://archive.ics.uci.edu/ml`

Table 6. Number of instances whose 1-nearest cluster centroid in SHC was at depth i (i = 1..18)

Dataset	recursion depth of the 1-nearest cluster centroid																	
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**
bl	0	11	14	27	61	12	0	0	0	0	0	0	0	0	0	0	0	0
bl10	0	0	1	22	49	46	6	1	0	0	0	0	0	0	0	0	0	0
bl30	0	0	0	18	70	33	3	1	0	0	0	0	0	0	0	0	0	0
bn	0	0	0	0	42	143	124	146	110	140	183	120	51	1	0	0	0	0
ecl	0	18	11	17	14	6	1	0	0	0	0	0	0	0	0	0	0	0
ecl10	0	10	31	19	7	0	0	0	0	0	0	0	0	0	0	0	0	0
ecl30	0	8	27	24	7	1	0	0	0	0	0	0	0	0	0	0	0	0
iris	10	7	6	2	5	0	0	0	0	0	0	0	0	0	0	0	0	0
lir	101	737	1595	1157	343	62	5	0	0	0	0	0	0	0	0	0	0	0
ls	98	103	53	338	229	235	189	41	1	0	0	0	0	0	0	0	0	0
ls10	0	4	26	362	656	226	13	0	0	0	0	0	0	0	0	0	0	0
ls30	0	0	5	229	768	274	11	0	0	0	0	0	0	0	0	0	0	0
mgt	0	0	0	0	29	52	46	74	108	321	600	915	926	509	175	48	1	0
mgt10	0	0	0	0	0	0	5	21	54	169	437	870	1016	680	388	132	30	2
pd	303	706	713	343	120	13	0	0	0	0	0	0	0	0	0	0	0	0
pd10	0	0	180	888	794	273	59	3	1	0	0	0	0	0	0	0	0	0
pd30	0	0	58	903	1036	180	20	1	0	0	0	0	0	0	0	0	0	0
ph	0	106	29	37	69	217	163	156	167	99	32	1	4	0	0	0	0	0
ph10	0	0	0	0	0	2	17	26	93	169	267	206	190	61	42	1	6	0
ph30	0	0	0	0	0		2	13	79	132	220	262	192	119	49	12	0	0
pm	0	0	0	0	0	23	44	41	38	6	1	0	0	0	0	0	0	0
pm10	0	0	0	0	4	8	25	45	43	12	16	0	0	0	0	0	0	0
pm30	0	0	0	0	3	8	27	34	29	40	12	0	0	0	0	0	0	0
sh	0	1808	2390	2998	1223	1673	605	613	197	69	23	0	0	0	0	0	0	0
tn	0	0	0	74	130	203	281	315	337	114	26	0	0	0	0	0	0	0
tn10	0	0	0	0	0	0	33	133	476	595	211	32	0	0	0	0	0	0
tn30	0	0	0	0	0	0	0	4	152	661	572	88	3	0	0	0	0	0
txr	168	170	331	341	75	15	0	0	0	0	0	0	0	0	0	0	0	0
txr10	0	7	256	556	244	34	3	0	0	0	0	0	0	0	0	0	0	0
txr30	0	1	136	713	221	27	2	0	0	0	0	0	0	0	0	0	0	0
ys	0	6	18	91	133	36	7	5	0	0	0	0	0	0	0	0	0	0
ys10	0	5	45	137	76	29	4	0	0	0	0	0	0	0	0	0	0	0
ys30	0	3	74	168	49	2	0	0	0	0	0	0	0	0	0	0	0	0

8. Ghosh, A.K.: On optimum choice of k in nearest neighbor classification. Comput. Stat. Data Anal. **50**(11), 31133123 (Jul 2006). https://doi.org/10.1016/j.csda.2005.06.007, https://doi.org/10.1016/j.csda.2005.06.007
9. Hand, D.J., Vinciotti, V.: Choosing k for two-class nearest neighbour classifiers with unbalanced classes. Pattern Recogn. Lett. **24**(910), 15551562 (Jun 2003). https://doi.org/10.1016/S0167-8655(02)00394-X, https://doi.org/10.1016/S0167-8655(02)00394-X
10. Johansson, U., Boström, H., König, R.: Extending nearest neighbor classification with spheres of confidence. In: Wilson, D., Lane, H.C. (eds.) Proceedings of the Twenty-First International Florida Artificial Intelligence Research Society Conference, May 15-17, 2008, Coconut Grove, Florida, USA. pp. 282–287. AAAI Press (2008), http://www.aaai.org/Library/FLAIRS/2008/flairs08-070.php
11. McQueen, J.: Some methods for classification and analysis of multivariate observations. In: Proc. of 5th Berkeley Symp. on Math. Statistics and Probability. pp. 281– 298. Berkeley, CA : University of California Press (1967)
12. Mullick, S.S., Datta, S., Das, S.: Adaptive learning-based k -nearest neighbor classifiers with resilience to class imbalance. IEEE Trans. Neural Networks Learn. Syst. **29**(11), 5713–5725 (2018). https://doi.org/10.1109/TNNLS.2018.2812279, https://doi.org/10.1109/TNNLS.2018.2812279
13. Ougiaroglou, S., Evangelidis, G.: RHC: a non-parametric cluster-based data reduction for efficient k-NN classification. Pattern Analysis and Applications **19**(1), 93–109 (2014). https://doi.org/10.1007/s10044-014-0393-7
14. Ougiaroglou, S., Nanopoulos, A., Papadopoulos, A.N., Manolopoulos, Y., Welzer-Druzovec, T.: Adaptive k-nearest-neighbor classification using a dynamic number of nearest neighbors. In: Proceedings of the 11th East European conference on Advances in databases and information systems. pp. 66–82. ADBIS'07, Springer-Verlag, Berlin, Heidelberg (2007), http://dl.acm.org/citation.cfm?id=1780119.1780129
15. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)