A fast hybrid *k*-NN classifier based on homogeneous clusters

Stefanos Ougiaroglou * and Georgios Evangelidis

Department of Applied Informatics, University of Macedonia
156 Egnatia St, 54006 Thessaloniki, Greece
{stoug,gevan}@uom.gr

Abstract. This paper proposes a hybrid method for fast and accurate Nearest Neighbor Classification. The method consists of a non-parametric cluster-based algorithm that produces a two-level speed-up data structure and a hybrid algorithm that accesses this structure to perform the classification. The proposed method was evaluated using eight real-life datasets and compared to four known speed-up methods. Experimental results show that the proposed method is fast and accurate, and, in addition, has low pre-processing computational cost.

Keywords: nearest neighbors, classification, clustering

1 Introduction

The k-Nearest Neighbor (k-NN) classifier [4] makes predictions by searching in the available Training Set (TS) for the k nearest items (neighbors) to a new item. The latter is assigned to the most common class among the retrieved k nearest neighbors. This method, in its simplest form, must compute all distances between the new item and all items in TS. Thus, the computational cost of searching depends on the size of TS and it may be prohibitive for large datasets and time-constrained applications.

The reduction of the cost of k-NN classifier remains an important open research issue that has attracted the interest of many researchers. Many methods have been proposed to speed-up k-NN searching. A possible categorization of these methods is: (i) Multi-attribute Indexes, (ii) Data Reduction Techniques (DRTs), and, (iii) Cluster-Based Methods (CBMs). DRTs, contrary to the other two categories, have the extra benefit of the reduction of storage requirements. The effectiveness of indexes [16] highly depends on the data dimensionality. In dimensions higher than ten, the curse of dimensionality may render their performance even worse than that of sequential search.

DRTs [18, 5, 20, 9, 6, 2, 13] reduce the computational cost of classification by building a small representative set of the initial training data, called the Condensing Set (CS). The idea behind DRTs is to apply the k-NN classifier over

* S. Ougiaroglou is supported by a scholarship from the Greek Scholarships Foundations (I.K.Y.)

this small set attempting to achieve accuracy as high as when using the original TS. Most DRTs produce their CS by keeping or generating for each class, many representative items (or prototypes) for the close-class-borders data areas and removing the items of the "internal" data areas.

DRTs can be divided into two main categories: (i) selection, and, (ii) abstraction algorithms. Both have the same motivation but differ on the way that they build the CS. Selection algorithms select some "real" TS items as prototypes. A typical example of this category is the well-known CNN-rule [7]. In contrast, abstraction algorithms generate prototypes by summarizing similar items. Examples of this category is the Chen and Jozwik method [3] and its variations (RSP algorithms [17]). Selection and abstraction algorithms are reviewed categorized and compared to each other in [5] and [18].

Contrary to DRTs, CBMs [8, 21, 10, 19] do not reduce the size of TS. They pre-process the training items and group them into clusters. For each new item, they dynamically form an appropriate training subset of the initial TS (or reference set) that is then used to classify the new item.

In our previous work [15], we demonstrated that DRTs and CBMs can be combined in a hybrid classification method to achieve the desirable performance. In particular, we proposed a pre-processing algorithm to construct a data structure and a fast algorithm to classify new items by accessing this structure. The main disadvantage of our method was that both algorithms were parametric and required a trial-and-error procedure to properly adjust their parameters.

Our motivation for this paper was the development of a non-parametric and fast nearest neighbor classification method for large and high dimensional data that combines two speed-up strategies, namely, DRTs and CBMs. We have extensively evaluated the proposed method and compared it to well-known DRTs and CBMs using eight real-life datasets through a cross-validation schema.

The rest of the paper is organized as follows. Section 2 considers in detail the proposed classification method. Section 3 presents the experimental results of its evaluation against other k-NN classification methods. Finally, Section 4 concludes the paper and gives some future directions.

2 The proposed method

The proposed classification method includes two major stages: (i) pre-processing, that is applied on the TS items in order to construct the Speed-up Data Structure (SUDS), and, (ii) classification, that uses the SUDS and applies the proposed hybrid classifier. In this section, we present the pre-processing algorithm as well as the hybrid classifier.

The pre-processing algorithm builds SUDS by finding homogeneous clusters in TS. A cluster is homogeneous if it contains items of a specific class only. The SUDS Construction Algorithm (SUDSCA) repetitively executes the well-known k-Means clustering algorithm [12] until all of the identified clusters become homogeneous. SUDS is a two-level data structure. Its first level is a list of centroids (or representatives) of the identified homogeneous clusters. Each one represents a

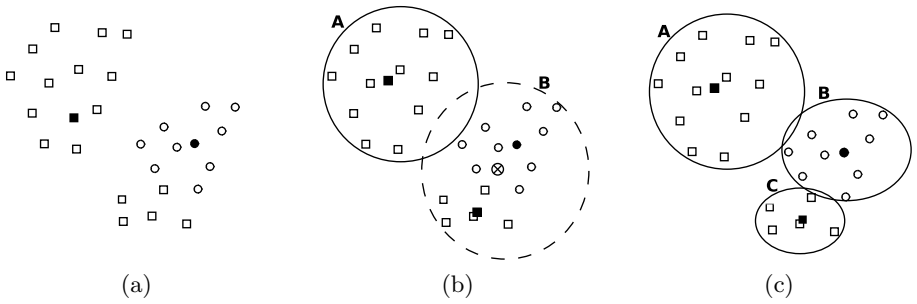


Fig. 1: SUDS construction by finding homogeneous clusters

data area of a specific class and indexes the "real" cluster items which are in the second level of SUDS. Figure 1 shows how SUDS is constructed and Algorithm 1 summarizes the steps of the corresponding algorithm.

Initially, SUDSCA finds the mean items (class-centroids) of each class in TS by averaging its items (Figure 1(a)). Then, it executes the k-Means clustering algorithm using these class centroids as initial means. Thus, for a dataset with M classes, SUDSCA initially identifies M clusters (Figure 1(b)). SUDSCA continues by analyzing the M clusters. If a cluster is homogeneous, it is added to SUDS. On the other hand, for each non-homogeneous cluster X, k-Means is executed on its items and identifies as many clusters as the number of distinct classes in X following the aforementioned procedure(Figure 1(c)). The repetitive execution of k-means terminates when all constructed clusters are homogeneous. Practically, SUDSCA constructs large clusters for internal class data areas, and small clusters for close-class-border data areas.

SUDSCA can be easily implemented using a simple queue data structure that stores the unprocessed clusters. Initially, the whole TS constitutes an unprocessed cluster and it becomes the head of the queue (line 1 in Algorithm 1). In each iteration, SUDSCA checks if cluster C in the head of the queue is homogeneous or not (line 4). If it is, the cluster is added to SUDS (lines 5-7). Otherwise, the algorithm computes a mean item for each class ($ClassCentroids$) present in C (lines 9-13). SUDSCA continues by calling the k-Means clustering algorithm for the items of C (line 14). This procedure returns a list of clusters ($NewClusters$) that are added to the queue structure (line 15) as unprocessed clusters. This procedure is repeated until the queue becomes empty (line 17).

Contrary to the pre-processing algorithm proposed in [15], SUDSCA is non-parametric. It determines the length of SUDS automatically (i.e., the number of clusters) based on the dataset used. SUDSCA extends on the idea of a previous work of ours that introduced a fast DRT called Reduction through Homogeneous Clusters (RHC) [14]. Here, our propose is not the development of a DRT, but the development of a hybrid, non-parametric method that combines DRTs and CBMs.

Algorithm 1 SUDS Construction Algorithm

Input: TS **Output:** $SUDS$

```
1: Enqueue(Queue, TS)
2: repeat
3:    C ← Dequeue(Queue)
4:    if C is Homogeneous then
5:       Compute the mean vector (class-centroid) M of C
6:       Put M into the first level of SUDS
7:       Put the items of C into the second level of SUDS and associate them to M
8:    else
9:       ClassCentroids ← ∅
10:      for each Class L in C do
11:         Centroid_L ← Compute the mean vector of items that belong to L
12:         ClassCentroids ← ClassCentroids ∪ Centroid_L
13:      end for
14:      NewClusters ← k-Means(C, ClassCentroids)
15:      for each cluster X in NewClusters do
16:         Enqueue(Queue, X)
17:      end for
18:   end if
19: until Queue is empty
20: return SUDS
```

The second part of the proposed method is a classifier that uses SUDS. It is called Hybrid Classification Algorithm based on Homogeneous Clusters (HCAHC) and is described in Algorithm 2. When a new item x arrives and must be classified (line 1 in Algorithm 2), HCAHC initially scans the first level of the SUDS and retrieves the Rk nearest representatives to x (lines 2-4). We call this scan a first level search. If all Rk retrieved representatives vote a specific class, x is classified to this class (lines 5-6). Otherwise, HCAHC goes to the second level of SUDS and x is classified by searching the k "real" items within the data subset dynamically formed by the union of the clusters of the Rk representatives (lines 8-10). We call this search a second level search.

Obviously, a second level search involves higher computational cost than a first level search. However, even in this case, HCAHC searches only a small subset of the initial TS data. For instance, suppose that SUDSCA has built a SUDS with 200 nodes and we have set Rk=8. HCAHC performs the first level search and retrieves the eight nearest representatives. Suppose that not all eight of them belong to the same class. As a result, HCAHC searches for the k nearest neighbors in the union of the eight clusters that correspond to the eight representatives and performs the classification. Even in this case, HCAHC avoids searching in the rest 192 clusters.

A new item can be classified via either a first or a second level search. Practically, the first level search is an abstraction DRT, while the second level search is a CBM. That is why HCAHC is a hybrid method. Furthermore, when HCAHC performs a second level search, it accesses an almost noise-free subset of

Algorithm 2 HCAHC Algorithm

Input: $SUDS$, Rk, k

```
for each new item x do
    Scan 1st level of SUDS and retrieve the Rk Nearest Representatives (NR) to x
    Find the majority class MC_1 of the Rk NR (ties are resolved by 1-NR)
    MCC ← COUNT(representatives of the majority class)
    if MCC = Rk then
        Classify x to MC_1
    else
        Scan within the set formed by the union of clusters of the Rk representatives
        and retrieve the k Nearest Neighbors (NNs) to x {Second level search}
        Find the majority class MC_2 of the k NNs (ties are resolved by 1-NN)
        Classify x to MC_2
    end if
end for
```

the initial TS. Since each cluster contains items of a specific class only, the subset (union of the Rk clusters) will not contain noisy items of other irrelevant classes, i.e., classes which are not represented by the Rk representatives. Thus, classification performance is not affected as much by noisy data. Of course, the length of SUDS depends on the level of noise. The more noisy items in TS, the higher the final number of homogeneous clusters (or length of SUDS).

Since we aim to a non-parametric method, we must find a way to automatically determine Rk. In the experiments of the following section, we have tested the effect of the value of Rk on the performance of our method. In addition, we use the empirical rule: $Rk = \lfloor\sqrt{|SUDS|}\rfloor$, where $|SUDS|$ is the number of nodes (clusters) in SUDS.

3 Performance Evaluation

3.1 Experimental Setup

The proposed classification method was tested using eight real life datasets distributed by KEEL Repository[1][1] (see Table 1). For comparison purposes, we evaluated: (i) CNN-rule [7], the first and one of the most popular selection DRTs, (ii) RSP3 [17], the well-known abstraction DRT, (iii) the CBM proposed by Hwang and Cho (Hwang's method) [8], and, (iv) our abstraction DRT called RHC [14]. All methods were implemented in C and evaluated using 5-fold cross validation. For each dataset, we used the five already constructed pairs of Training/Testing sets hosted by KEEL repository. These sets are appropriate for 5-fold cross validation. Furthermore, we used the Euclidean distance as the distance metric.

CNN-rule, RSP3 and RHC are non-parametric methods, that is, they do not use user-defined parameters in order to reduce the data. On the other hand,

[1] http://sci2s.ugr.es/keel/datasets.php

Table 1: Dataset description - Conventional k-NN Classifier performance

Dataset	Size	Attr.	Classes	Conv-k-NN	
				Acc. (%)	Cost (M)
Letter Recognition (LR)	20000	16	26	96.01	64.00
Magic Gamma Telescope (MGT)	19020	10	2	99.37	19.34
Pen-Digits (PD)	10992	16	10	91.22	6.63
Landsat Satellite (LS)	6435	36	6	81.32	57.88
Shuttle (SH)	58000	9	7	99.82	538.24
Texture (TXR)	5500	40	11	99.02	4.84
Phoneme (PH)	5404	5	2	90.10	4.67
Ring (RNG)	7400	20	2	74.69	8.76

Hwang's method is parametric. In addition to parameter k (number of nearest neighbors to search), which is used by all methods during the classification step, it uses three extra parameters. For two of those parameters we used the values suggested by Hwang and Cho in their experiments. The third parameter, C, is used during the pre-processing phase and is related to the number of clusters that are constructed by the k-Means clustering algorithm. We built eight Hwang's classifiers using different C values. More specifically, each classifier i=1,...,8, used $C = \lfloor \sqrt{\frac{n}{2^i}} \rfloor$, where n is the number of TS items. The first classifier, i.e. i=1 is based on the rule of thumb $C = \lfloor \sqrt{\frac{n}{2}} \rfloor$ [11].

Although SUDSCA is non-parametric, HCAHC is a parametric classifier. In addition to k, it uses parameter Rk. We built 29 HCAHC classifiers, for Rk=2,3,...,30, and we also considered the automatic determination of Rk (see Section 2). We refer to that classifier as HCAHC-sqrt. For both HCAHC and Hwang's method, we report only the most accurate classifiers for each cost.

During the classification step, all methods involve parameter k. The DRTs perform k-NN classification using their CS, while Hwang's method does this over a small reference set that is dynamically formed for each new item. Finally, HCAHC searches for k nearest neighbors when it performs a second level search. We used the best k values for each method and dataset, i.e., the value that achieved the highest classification accuracy. In effect, we ran the cross validation many times for different k values and kept the best one.

3.2 Pre-processing comparisons

Table 2 presents the pre-processing computational costs in terms of millions (M) distance computations (how many distances were computed during the pre-processing). As we expected, SUDSCA and RHC were executed very fast in comparison to the other approaches. This happened because: (i) the construction of SUDS and of the RHC condensing set are based on the repetitive execution of the fast k-Means clustering algorithm, and, (ii) in both cases, k-Means uses the mean items of the classes as initial centroids, and thus, clusters are consolidated very quickly. It is worth mentioning that SUDSCA and RHC pre-processing

Table 2: Preprocessing Cost (in million of distance computations)

Dataset	CNN	RSP3	Hwang's method				RHC/
			i=1	i=3	i=5	i=7	SUDSCA
LR	163.03	326.52	88.88	63.66	26.35	10.89	41.85
MGT	277.88	511.67	142.99	80.62	21.10	12.83	4.09
PD	11.76	94.80	28.80	11.27	5.97	1.70	2.88
LS	18.59	37.70	16.74	12.44	4.54	0.81	1.69
SH	45.40	17597.68	744.82	399.23	105.13	34.78	16.83
TXR	5.57	27.63	14.86	7.43	3.89	0.83	3.63
PH	13.47	20.32	9.87	3.70	1.33	0.74	0.65
RNG	29.63	43.42	18.48	12.35	5.50	2.83	2.00
Avg.	70.58	2332.47	133.18	73.84	21.73	8.18	9.20

could become even faster had we used a different k-Means stopping criterion than the full clusters consolidation (no item move from one cluster to another during a complete algorithm pass).

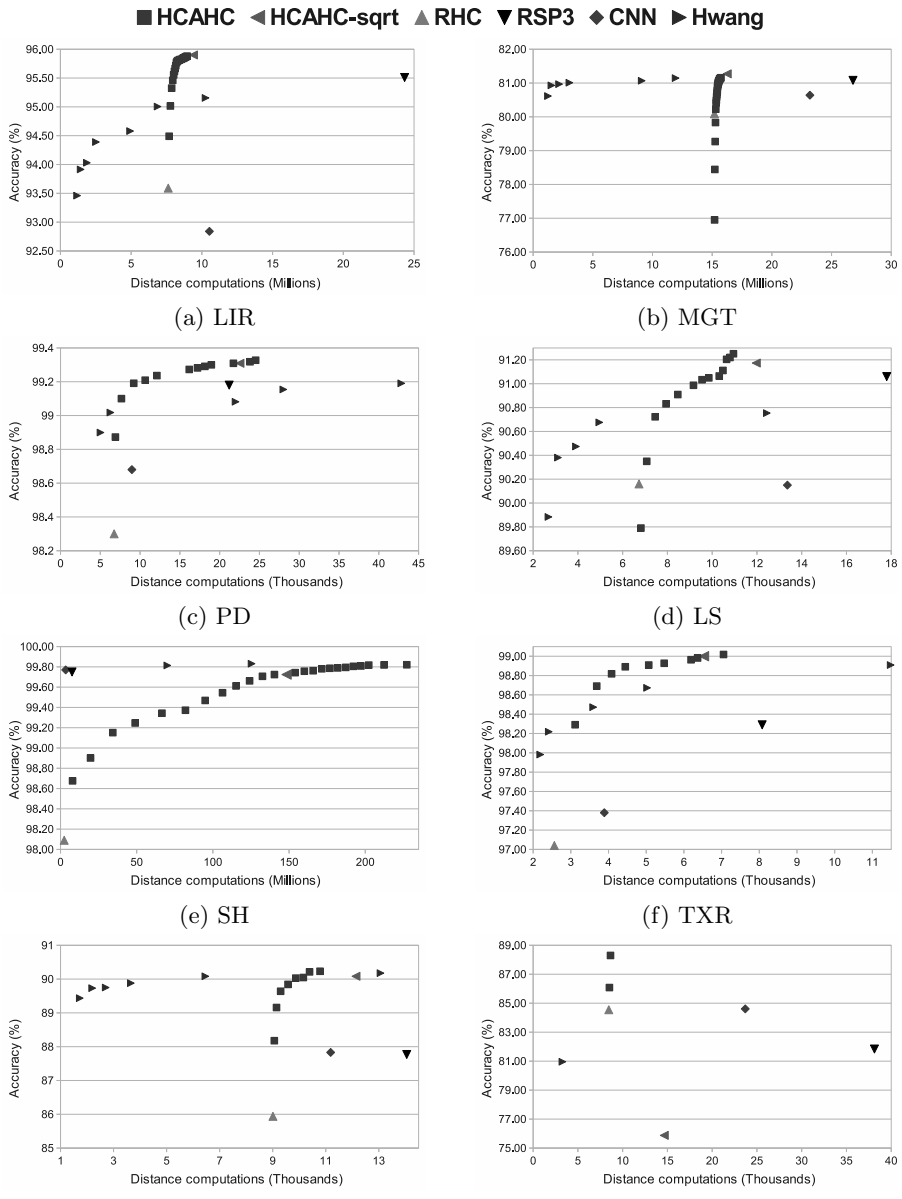
Concerning the other methods, RSP3 was the most time consuming approach. This is because RSP3 repetitively executes a costly procedure for finding the most distant items in data groups. Hwang's method for $i \geq 5$ is executed very fast. However, in real applications, the user must perform a trial-end-error procedure for determining the parameters. This may render pre-processing a hard and extremely time consuming procedure. Although CNN-rule is quite faster than RSP3, its pre-processing cost remains at high levels.

3.3 Classification performance comparisons

We performed the classification step by using seven classification methods on the eight datasets without any previous knowledge about the data (like distribution of classes, shape and size of the class data areas). The methods used were: (i) Conventional k-NN (conv-k-NN), (ii) HCAHC, (iii) HCAHC-sqrt, (iv) RHC, (v) CNN, (vi) RSP3, and, (vii) Hwang's method.

The performance measurements of conv-k-NN are shown in Table 1 while the measurements of the speed-up methods are depicted in Figure 2. In particular, figure 2 includes one diagram for each dataset. The eight diagrams present the cost measurements (in terms of millions or thousands distance computations) on the x-axis and the corresponding accuracy on the y-axis. The cost measurements indicate how many distances were computed in order to classify all testing items. Since, we used a cross-validation schema, cost measurements are average values.

Almost in all cases, HCAHC and HCAHC-sqrt achieved noteworthy performances. In some cases, they even reached the accuracy level of conv-k-NN. All diagrams show that rule $Rk = \lfloor \sqrt{|SUDS|} \rfloor$ is a good choice for the determination of Rk. With the exception of the SH dataset, HCAHC achieved better classification performance than all DRTs. On the other hand, although HCAHC and HCAHC-sqrt achieved higher accuracy than Hwang's method in all datasets,



(a) LIR (b) MGT

(c) PD (d) LS

(e) SH (f) TXR

(g) PH (h) RNG

Fig. 2: Classification performance (Accuracy vs Computational cost)

for the MGT, SH, and PH datasets, Hwang's method may be preferable because it achieved accuracies close to those of HCAHC and HCAHC-sqrt at a lower computational cost.

Contrary to all other datasets, RNG was the only dataset where low Rk parameter values performed better than higher values (in Figure 2(h), HCAHC classifiers were built with Rk=2 and Rk=3. HCAHC-sqrt was built using a higher Rk value. Consequently, the corresponding performance measurements were quite bad.

Concerning the SH dataset, CNN and RSP3 built very small Condensing Sets. Thus, the k-NN classifiers that executed over these CSs, were not only accurate but very fast as well. HCAHC and HCAHC-sqrt were able to achieve the accuracy levels of CNN and RSP3 at a higher computational cost. Finally, we should consider the fact that SH is an imbalanced dataset. It has two very rare classes. Despite that, HCAHC and HCAHC-sqrt were able to classify testing items of the rare classes with high accuracy.

4 Conclusions

Speeding-up distance based classifiers is a very important issue in data mining. In this paper, we presented and evaluated a hybrid classification method. The motivation of our work was the development of a non-parametric method that has low pre-processing cost and is able to classify new items fast and with high accuracy. We presented a non-parametric fast pre-processing algorithm that builds a two-level data structure, and a hybrid classifier that makes predictions by accessing either the first or the second level of this structure. The proposed classifier is parametric since it uses parameter Rk (number of cluster representatives to use in a first level search). However, we demonstrated how Rk can be automatically determined and render the proposed method non-parametric. Experimental results based on eight real-life datasets showed that our method achieved the aforementioned goals.

Our future work includes the incremental execution of the clustering pre-processing procedure. Although the pre-processing algorithm has low computational cost, rebuilding the structure from scratch may be inadequate for dynamic environments.

References

1. Alcalá-Fdez, J., Sánchez, L., García, S., del Jesús, M.J., Ventura, S., i Guiu, J.M.G., Otero, J., Romero, C., Bacardit, J., Rivas, V.M., Fernández, J.C., Herrera, F.: Keel: a software tool to assess evolutionary algorithms for data mining problems. Soft Comput. 13(3), 307–318 (2009)
2. Brighton, H., Mellish, C.: Advances in instance selection for instance-based learning algorithms. Data Min. Knowl. Discov. 6(2), 153–172 (Apr 2002)
3. Chen, C.H., Jóźwik, A.: A sample set condensation algorithm for the class sensitive artificial neural network. Pattern Recogn. Lett. 17, 819–823 (July 1996)

4. Dasarathy, B.V.: Nearest neighbor (NN) norms : NN pattern classification techniques. IEEE Computer Society Press (1991)
5. Garcia, S., Derrac, J., Cano, J., Herrera, F.: Prototype selection for nearest neighbor classification: Taxonomy and empirical study. IEEE Transactions on Pattern Analysis and Machine Intelligence 34(3), 417–435 (2012)
6. Grochowski, M., Jankowski, N.: Comparison of instance selection algorithms ii. results and comments. In: Artificial Intelligence and Soft Computing - ICAISC 2004, Lecture Notes in Computer Science, vol. 3070, pp. 580–585. Springer Berlin / Heidelberg (2004)
7. Hart, P.E.: The condensed nearest neighbor rule. IEEE Transactions on Information Theory 14(3), 515–516 (1968)
8. Hwang, S., Cho, S.: Clustering-based reference set reduction for k-nearest neighbor. In: 4th international symposium on Neural Networks: Part II–Advances in Neural Networks. pp. 880–888. ISNN '07, Springer (2007)
9. Jankowski, N., Grochowski, M.: Comparison of instances seletion algorithms i. algorithms survey. In: Artificial Intelligence and Soft Computing - ICAISC 2004, Lecture Notes in Computer Science, vol. 3070, pp. 598–603. Springer Berlin / Heidelberg (2004)
10. Karamitopoulos, L., Evangelidis, G.: Cluster-based similarity search in time series. In: Proceedings of the Fourth Balkan Conference in Informatics. pp. 113–118. BCI '09, IEEE Computer Society, Washington, DC, USA (2009)
11. Mardia, K., Kent, J., Bibby, J.: Multivariate Analysis. Academic Press (1979)
12. 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)
13. Olvera-López, J.A., Carrasco-Ochoa, J.A., Martínez-Trinidad, J.F., Kittler, J.: A review of instance selection methods. Artif. Intell. Rev. 34(2), 133–143 (Aug 2010)
14. Ougiaroglou, S., Evangelidis, G.: Efficient data-set size reduction by finding homogeneous clusters. In: Procendings of the fifth Balkan Conference in Informatics. p. to appear. BCI '12, ACM (2012)
15. Ougiaroglou, S., Evangelidis, G., Dervos, D.: An adaptive hybrid and cluster-based model for speeding up the k-nn classifier. In: Hybrid Artificial Intelligent Systems, Lecture Notes in Computer Science, vol. 7209, pp. 163–175. Springer Berlin / Heidelberg (2012)
16. Samet, H.: Foundations of multidimensional and metric data structures. The Morgan Kaufmann series in computer graphics, Elsevier/Morgan Kaufmann (2006)
17. Sánchez, J.S.: High training set size reduction by space partitioning and prototype abstraction. Pattern Recognition 37(7), 1561–1564 (2004)
18. Triguero, I., Derrac, J., García, S., Herrera, F.: A taxonomy and experimental study on prototype generation for nearest neighbor classification. IEEE Transactions on Systems, Man, and Cybernetics, Part C 42(1), 86–100 (2012)
19. Wang, X.: A fast exact k-nearest neighbors algorithm for high dimensional search using k-means clustering and triangle inequality. In: The 2011 International Joint Conference on Neural Networks (IJCNN). pp. 1293 –1299 (August 2011)
20. Wilson, D.R., Martinez, T.R.: Reduction techniques for instance-based learning algorithms. Machine Learning 38(3), 257–286 (2000)
21. Zhang, B., Srihari, S.N.: Fast k-nearest neighbor classification using cluster-based trees. IEEE Trans. Pattern Anal. Mach. Intell. 26(4), 525–528 (2004)