An Adaptive Hybrid and Cluster-Based Model for speeding up the *k*-NN Classifier

Stefanos Ougiaroglou*[1], Georgios Evangelidis[1], and Dimitris A. Dervos[2]

[1] Dept. of Applied Informatics, University of Macedonia, 54006 Thessaloniki, Greece
{stoug, gevan}@uom.gr
[2] Dept. of Informatics, Alexander TEI of Thessaloniki, 57400 Sindos, Greece
dad@it.teithe.gr

Abstract. A well known classification method is the k-Nearest Neighbors (k-NN) classifier. However, sequentially searching for the nearest neighbors in large datasets downgrades its performance because of the high computational cost involved. This paper proposes a cluster-based classification model for speeding up the k-NN classifier. The model aims to reduce the cost as much as possible and to maintain the classification accuracy at a high level. It consists of a simple data structure and a hybrid, adaptive algorithm that accesses this structure. Initially, a preprocessing clustering procedure builds the data structure. Then, the proposed algorithm, based on user-defined acceptance criteria, attempts to classify an incoming item using the nearest cluster centroids. Upon failure, the incoming item is classified by searching for the k nearest neighbors within specific clusters. The proposed approach was tested on five real life datasets. The results show that it can be used either to achieve a high accuracy with gains in cost or to reduce the cost at a minimum level with slightly lower accuracy.

Keywords: *k*-NN classification, cluster-based classification, data reduction

1 Introduction

The *k*-Nearest Neighbors (*k*-NN) classification algorithm is a widely-used lazy classifier [2]. When a new item needs to be classified, *k*-NN searches the available data (training set) and retrieves the k nearest neighbors to it according to a distance metric (e.g., euclidean distance). The new item is classified to the class that is the most common one among the classes of the retrieved k nearest neighbors, with possible ties resolved either randomly or by choosing the class of the nearest neighbor.

The *k*-NN classifier is considered to be an effective classifier and has many applications. Its main drawback is that the computational cost needed to compute all distances between a new item and the training data can be very high.

* S. Ougiaroglou is supported by the State Scholarship Foundation of Greece (I.K.Y.).

The research conducted on the reduction of the k-NN cost is based on indexing, data reduction and cluster-based methods.

Multi-attribute indexing methods can speed up nearest neighbor searches [11]. They are very effective when datasets have moderate dimensionality (e.g. 2-10). In higher dimensions, the phenomenon of "dimensionality curse" renders those indexes irrelevant since their performance degrades rapidly and can become worse than that of the sequential scan.

Data Reduction Techniques (DRTs) [14, 4, 7, 15, 13] build a small representative set of the initial data, often called Condensing Set (CS). The idea is to apply k-NN on this set attempting to achieve almost the same accuracy as with the original data at a much lower cost. These methods can be divided into two main categories: (i) filtering (or selection) [4], and (ii) abstraction (or generation) [14] algorithms. Filtering algorithms select items from the Training Set (TRS) as representatives. On the other hand, abstraction algorithms generate representatives by summarizing similar items.

Finally, Cluster-Based Methods (CBM) preprocess the TRS items and place them into clusters [6, 17]. When a new item must be classified, they dynamically decide which subset of the training data will be used. Contrary to DRTs, CBMs and indexing methods do not reduce the storage requirements.

Our motivation is to address the problem of classifying large and high-dimensional datasets, where dimensionality reduction negatively affects the accuracy, thus, indexing is not applicable. We combine two strategies, abstraction and clustering, in a hybrid classification schema to speed-up the k-NN classifier. First, a clustering preprocessing task builds a two level data structure. Its first level contains cluster centroids (representatives) for each class, and, its second level contains the set of items belonging to each such cluster. Then, an adaptive and hybrid algorithm attempts to achieve high accuracy while keeping the computational cost as low as possible. A new item is classified either using the first or the second level of the data structure. So, an abstraction and a cluster-based approach are combined to achieve the desirable performance.

The rest of this paper is organized as follows. Section 2 briefly presents the related work, and Section 3 considers in detail the proposed classification model. In Section 4, experimental results based on real life datasets are presented, and the paper concludes in Section 5.

2 Related Work

One of the most widely used filtering algorithms is the Condensing Nearest Neighbor (CNN) rule [5]. It reduces the cost of k-NN by removing the non close-border items. The idea is that these items can be removed without significant loss of accuracy. The CNN-rule determines the amount of the selected representative items automatically based on the level of noise in the data and the number of classes. Many other algorithms either extend the CNN-rule or are based on the same idea. However, the CNN-rule algorithm continues to be the reference algorithm and it is used in many works for comparison purposes.

A recently proposed filtering algorithm is the Prototype Selection by Clustering (PSC) [10]. PSC is based on the idea that homogeneous clusters (not containing items that belong to different classes) include items that lie in the "internal" area of a class, whereas, non-homogeneous clusters include close-border items. PSC uses k-means clustering in order to divide the TRS into clusters (any clustering method can be used). Then, for each homogeneous cluster, the nearest item to the cluster centroid is placed into the CS, whereas, for each non-homogeneous cluster, only the items that define the decision boundaries are placed into the CS.

Chen and Jozwik proposed a well-known abstraction algorithm [1]. Chen's algorithm is based on dividing the TRS into small subsets. The algorithm begins by finding the pair of the most distant points, A, B, in the TRS. It continues by splitting the TRS into two subsets. One subset contains the items nearest to A, and, the other the items nearest to B. Then, it selects to split the subset with the greatest diameter. This procedure continues until the number of subsets becomes equal to the user-predefined CS size. Finally, Chen's algorithm computes a mean item for each subset. The class of each mean item is defined to be the most common class in the corresponding subset.

Sanchez introduced three Reduction by Space Partitioning (RSP) algorithms that are based on the idea of Chen's algorithm [12]. Contrary to Chen's algorithm, RSP1 computes as many mean items as the number of different classes in each subset. RSP1 and RSP2 differ on the way that they select the subset that will be divided. RSP3 continues splitting until all subsets are homogeneous. Thus, contrary to the other RSP methods, RSP3 determines the CS size automatically.

Editing approaches constitute a subcategory of filtering algorithms. Their goal is to increase the accuracy rather than reduce the cost. This is achieved by removing noisy and close-border data, leaving smoother decision boundaries. The reduction rates of many filtering/abstraction algorithms depend on the level of noise that exists in the data. Thus, in many classification tasks, an editing algorithm is used to remove the noisy data before the application of the main reduction procedure [7]. However, some hybrid filtering approaches, such as the DROP algorithms [15], integrate the idea of editing. They build the CS and simultaneously remove noisy items (see [4] for details). A well known editing algorithm is the Edited Nearest Neighbor (ENN) rule [16]. For each TRS item x, if the class of x does not agree with the majority of its k nearest neighbors, x is removed. ENN-rule needs to compute $\frac{N*(N-1)}{2}$ distances, i.e., all the distances among the TRS items.

Many other Abstraction and Filtering algorithms are reviewed, categorized, evaluated and compared to each other in [14] and [4]. Other relevant reviews can be found in [7, 13, 15].

Hwang and Cho have proposed an effective CBM [6]. It uses the k-means algorithm [9] to find clusters in the data. Then, each cluster is divided into two sets. Items located in a certain distance from the cluster centroid are placed into the "core set", while the rest are placed into the "peripheral set". If a new item

lies within the "core area" of the nearest cluster, it is classified by retrieving the k-nearest neighbors from this cluster. Otherwise, the nearest neighbors are retrieved from the set formed by the items of the nearest cluster and the "peripheral" items of adjacent clusters. Another effective CBM is the Cluster-based Tree [17]. It is based on searching in a cluster hierarchy and can be used for either metric or non-metric spaces.

3 The Proposed Classification Model

The proposed model consists of two parts: (i) a Speed-Up Data Structure (SUDS) built by a clustering preprocessing procedure, and, (ii) a Hybrid and Adaptive Classification Algorithm that uses this structure.

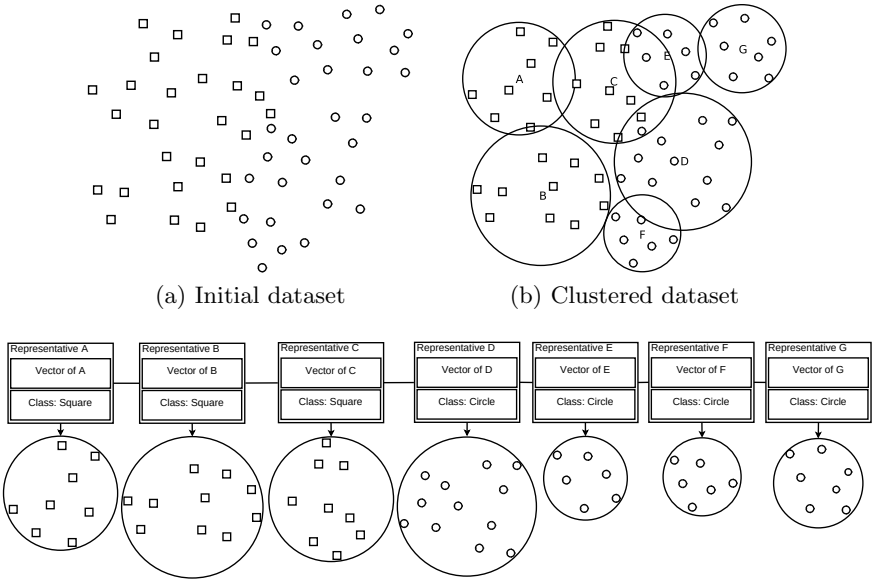
3.1 Speed-up data structure construction

Initially, the training data is preprocessed by the k-means algorithm to form the data structure (SUDS). More specifically, for each class, k-means identifies a number of clusters. SUDS consists of two data levels. The first level is a list of representatives (the mean vectors of all clusters for all classes). Each node of this list points to a list that contains the items assigned to the specific representative. These lists of items form the second level of SUDS.

We use a parameter, the Data Reduction Factor (DRF), to determine the number of clusters that will be created. For each class C, the number of clusters NC is estimated by $NC = \lceil \frac{|C|}{DRF} \rceil$, where $|C|$ is the number of items that belong to C. Thus, the DRF parameter specifies the number of the clusters that will be created. Figure 1 illustrates a two-dimensional example. In this case there are two classes, square and circle. The initial dataset includes 27 squares and 31 circles (Figure 1(a)). Thus, if DRF is set to 10, the classes Square and Circle should be represented by 3 and 4 mean vectors respectively (Figure 1(b)). The result of the clustering procedure will be the two level SUDS depicted in Figure 1(c). Class square is represented by the mean vectors A–C, and class circle by D–G.

3.2 The Fast Hybrid and Adaptive Classification Algorithm

The second part of the model comprises a Fast Hybrid and Adaptive Classification Algorithm (FHACA) that accesses the SUDS (Algorithm 1). FHACA uses three (input) parameters that let the user define the desirable trade-off between accuracy and cost. The idea behind the algorithm is quite simple. When a new item x has to be classified, FHACA initially scans the first level of SUDS (first level search) and retrieves the Rk nearest representatives to x. If the acceptance criterion introduced by the $NRRatio$ parameter is met, these representatives determine the class where x belongs to. Upon failure, x is classified by searching for the k "real" nearest neighbors within the clusters of the Rk nearest representatives (second level search). Obviously, the more items classified without the need of the second level search, the lower is the computational cost involved.



(a) Initial dataset (b) Clustered dataset

(c) Speed-up data structure

Fig. 1: k-means clustering on items of each class (DRF=10)

FHACA uses the $NRRatio$ parameter to decide when to switch to a second level search. This parameter defines how many nearest representatives should determine the majority class (the most common class among the Rk nearest representatives) in order to classify the new item. For instance, suppose that the input parameters are set to be k=3, Rk=10, and $NRRatio$=70%. Also, suppose that a new item x should be classified. FHACA, initially, retrieves and examines the 10 nearest representatives from the first level of SUDS. If 7 or more of them belong to the same class, then x is classified to this class. Otherwise, the 3 "real" nearest neighbors are retrieved from the data subset formed by the union of clusters (second level of SUDS) of the 10 nearest representatives, and they determine the class of x.

3.3 Discussion

Considering the proposed classification algorithm, it is obvious that an unclassified item that lies in a close-border area, is classified by a second level search. On the other hand, an item that lies in the "internal" area of a class, is classified by the mean vectors (first level search). Thus, the proposed method is neither a cluster-based nor an abstraction method, since it dynamically decides on how a new item is classified. The first level comprises an abstraction method that uses the mean vectors obtained by the clustering preprocessing procedure. On the other hand, the second level comprises a cluster-based method that uses a

Algorithm 1 The Fast Hybrd and Adaptive Classification Algorithm

Input: *SUDS, k, Rk, NRRatio*

```
1: for each unclassified item x do
2:    Scan 1st level of SUDS and retrieve the Rk Nearest Representatives (NRs) to x
3:    Find the majority class MC_1 of the Rk NRs (ties are resolved by 1-NR)
4:    MCCounter ← COUNT(representatives of the majority class)
5:    if MCCounter ≥ NRRatio then
6:       Classify x to MC_1
7:    else
8:       Scan within the set formed by the union of the clusters of the Rk representa-
         tives and retrieve the k Nearest Neighbors (NNs) to x
9:       Find the majority class MC_2 of the k NNs (ties are resolved by 1-NN)
10:      Classify x to MC_2
11:   end if
12: end for
```

dynamically-formed subset of the initial TRS. Hence, the method is a hybrid approach. Moreover, it differs from the DRTs in that, as in the cases of indexing and CBMs, it does not reduce the storage requirements.

Concerning the parameters of the proposed model, Rk and $NRRatio$ should be determined by taking into account the DRF value that was used for SUDS construction. If accuracy is more critical than cost and a SUDS with few and large clusters is available, Rk and $NRRatio$ should have high values. On the other hand, if cost is more critical and a SUDS with many and small clusters is available, low Rk and $NRRatio$ values are recommended. Considering the DRF value, low DRF values are recommended for building accurate classifiers with high cost savings and high DRF values for building fast classifiers without significant accuracy loss. If our needs are not specified at the time that SUDS is constructed, an intermediate DRF value is the most appropriate. In this case, the trade-off can be afterwards determined by adjusting Rk and $NRRatio$.

Furthermore, when FHACA executes a second level search, it accesses a subset of the initial TRS formed by the union of the Rk clusters. Since each cluster includes items of a specific class, this dataset is an almost noise free dataset (it does not include noisy items of classes which are not represented by the Rk representatives) and, thus, the classification accuracy is not affected as much by noisy data. Taking into account this property, editing is not as necessary as in many filtering/abstraction algorithms. Of course, noise removal and overlapping "cleaning" among regions of different classes could increase the cluster quality and the overall classification performance.

4 Performance Evaluation

The proposed model was implemented in C and evaluated using five real life datasets distributed by the UCI Repository [3]. They are summarized in Table 1. All datasets were used without data normalization or any other transformation.

Table 1: Dataset description (cost is in million distance computations)

Dataset	**Train/Test dataset size**	**Attr.**	**Classes**	**Best** k	**Accuracy** (%)	**Cost** (M)
Letter Recognition(LR)	15000/5000	16	26	4	95.68	75
Magic Gamma Telescope(MG)	14000/5020	10	2	12	81.39	70.28
Pendigits(PD)	7494/3498	16	10	4	97.89	26.21
Landsat Satellite(LS)	4435/2000	36	6	4	90.75	8.87
Shuttle(SH)	43500/14500	9	7	1	99.88	630.75

The computational cost measurements were estimated by counting the distance computations required to classify all items of the Testing Set (TES) by scanning the training data. All distances were estimated using the Euclidean distance metric. Since the conventional k-NN classifier (conv-k-NN) requires the computation of all distances between each TES item and the initial TRS items, its cost can be estimated by multiplying the cardinalities of the training and testing sets (see the second and last column of Table 1). For instance, conv-k-NN computes $15000 \times 5000 = 75$M distances for LR. The fifth column lists the k value found to achieve the highest accuracy. For comparison purposes, we implemented in C two filtering, an abstraction, and, a cluster-based approach. We selected CNN-rule [5], PSC [10], RSP3 [12] and Hwang's algorithm [6], respectively.

4.1 Experimental Setup

The adaptive schema of the proposed model provides four parameters: DRF, Rk, $NRRatio$, and, k. We defined k to be the best k value of conv-k-NN (Table 1). Several experiments were conducted for the other parameters. The values tested for each one were: (a) DRF 4,6,8,10,20,30,...,300, (b) Rk 1,2,...,30, and (c) $NRRatio$ 51%, 70% and 100%. Thus, for each dataset, we built and evaluated 2970 ($33 \times 30 \times 3$) FHACA classifiers. In the end, we kept the most accurate FHACA classifier for each reported cost. In real life applications, there is no need to do such extensive tests to determine the appropriate values of the parameters. Here, our purpose was to fully understand how each parameter influences the model construction and its performance. In real life applications, the parameters should be determined by taking into consideration the accuracy and cost significance as well as the dataset used.

Hwang's algorithm also uses four parameters: C is the number of clusters, L is the number of adjacent clusters, D is the threshold that defines the core and peripheral items, and, k defines the number of nearest neighbors (see Section 2). We set $L = \lfloor\sqrt{k}\rfloor$ as Hwang and Cho did in their experiments. We set $C = \lfloor\sqrt{\frac{n}{2^i}}\rfloor$, i=1,...,8, where n is the number of items. Thus, for each dataset, we built 8 classifiers. The first classifier (for i=1) is based on the rule of thumb that defines $C = \lfloor\sqrt{\frac{n}{2}}\rfloor$ [8]. We decided to build classifiers that use small C values based on the observation that Hwang and Cho defined C=10 for a TRS with

60919 items (they did not find the optimal C value). Of course, the fewer and larger the clusters, the higher the cost of the classifiers (see Fig. 2–6). Following the approach of Hwang and Cho, we considered as peripheral items, those whose distance from the cluster centroid was greater than the double average distance among the items of each cluster (i.e. D=2). Finally, we chose the k values that achieved the highest accuracy.

Another issue that needs attention is the number of clusters that PSC uses. In [10], the authors executed experiments by constructing $r \times j$, $j = 2, 4, \cdots, 10$, clusters, where r is the number of discrete classes in a dataset. Since our main goal is to achieve high accuracy at a low cost, we decided to test PSC with higher j values. We conducted several experiments with varying j values (up to 200 and for the noisy MG dataset up to 2000).

Concerning CNN-rule, PSC and RSP3, two experiments were conducted for each one, one on the original and one on the edited TRS. For editing purposes, we implemented the ENN-rule [16] and used it by setting k=1 for all datasets. Thus, each method was tested with two k-NN classifiers, one for each condensing set. We refer to them as CNN-kNN, ENN-CNN-kNN, RSP3-kNN, ENN-RSP3-kNN, PSC-kNN and ENN-PSC-kNN. The k parameter for classifiers was adjusted to achieve the highest accuracy.

Since only the first level search (FHACA-1st LS) can be used to classify new items, we present its performance in the diagrams of subsection 4.2 (Fig. 2–6). FHACA-1st LS carries out the whole classification task when $NRRatio$ is set to zero. In other words, the k-NN classifier classifies the new data using only the set of representatives produced by the k-means algorithm.

4.2 Comparisons

Table 2 presents a small subset of preprocessing measurements for each method. Specifically, it includes the number of representatives/clusters as well as the preprocessing computational costs required by each method (the cost of editing is not included). The very high preprocessing cost of RSP3 is the result of the farthest point computations in the subsets. Since the preprocessing is executed only once, these cost measurements may be not so significant. However, they have to be evaluated taking into account the performance that the corresponding classifiers achieve.

We now focus on the accuracy-cost measurements obtained by executing the classifiers on the five datasets. In the following, we will be reporting the FHACA parameter values in parenthesis in this order (DRF, Rk, $NRRatio$). Figures 2 through 6 report, for each classifier, the cost measurements on the x-axis and the corresponding accuracy values on the y-axis.

Letter Recognition (Figure 2): FHACA achieved accuracy between 94%–96% with the lowest cost. For instance, an effective FHACA classifier achieved an accuracy of 95.52% with 4.1M computations (30,10,100). All other methods had higher cost and achieved lower accuracy. The first three Hwang classifiers (i=1–3) may be preferable for accuracy levels between 93%–94%. The highest FHACA accuracy was 96% (8,27,100).

Table 2: Preprocessing

Method		LR	MG	PEN	LS	SH
CNN-rule	Items:	2517	5689	312	909	300
	Cost:	145,386,010	217,900,759	7,940,953	13,545,272	57,958,973
RSP3	Items:	5906	6646	857	1219	688
	Cost:	291,151,380	412,752,916	70,561,629	28,929,950	15,671,718,080
PSC	Items.:	3075	4024	362	689	591
r=10	Cost:	187,371,957	17,796,445	20,254,707	6,970,603	259,638,105
PSC	Items.:	2813	3868	582	746	633
r=50	Cost:	429,018,237	122,380,296	86,182,235	30,608,866	1,027,702,841
PSC	Items.:	3583	3838	1056	1000	664
r=100	Cost:	390,007,005	187,853,770	112,410,384	42,579,896	1,777,429,582
Hwang	Clust.:	86	83	61	47	147
i=1	Cost:	56,778,655	235,903,403	18,294,684	21,683,796	556,375,731
Hwang	Clust.:	43	41	30	23	73
i=3	Cost:	26,460,903	46,508,820	9,900,009	4,492,908	400,159,128
Hwang	Clust.:	21	20	15	11	36
i=5	Cost:	21,750,210	9,534,190	3,155,079	3,760,935	128,456,130
Hwang	Clust.:	10	10	7	5	18
i=7	Cost:	11,715,045	11,634,045	951,759	691,870	36,061,653
SUDS	Clust.:	3769	3502	1880	1112	10880
DRF=4	Cost:	12,832,249	243,430,728	9,922,935	6,679,461	4,372,102,123
SUDS	Clust.:	1515	1402	755	447	4353
DRF=10	Cost:	8,907,558	273,815,604	6,220,083	4,876,310	2,910,000,719
SUDS	Clust.:	763	702	380	224	2178
DRF=20	Cost:	5,585,990	161,018,546	4,356,282	2,976,219	1,929,776,967
SUDS	Clust.:	389	352	192	113	1091
DRF=40	Cost:	3,569,682	87,023,904	3,558,287	2,377,213	2,032,629,026
SUDS	Clust.:	160	141	80	47	440
DRF=100	Cost:	1,520,838	51,632,480	1,553,272	1,285,263	1,462,204,134
SUDS	Clust.:	82	71	40	26	222
DRF=200	Cost:	671,965	26,615,055	368,204	478,202	972,749,420

Magic Gamma Telescope (Figure 3): Again, FHACA achieved high accuracy (over 81%) having lower cost than all other methods. For instance, FHACA achieved an accuracy of 81,39% requiring 6.26M computations (40,19,100). A "cheeper" FHACA classifier required 2.62M computations for an accuracy of 81.14% (50,5,100). The highest FHACA accuracy (81.47%) was achieved with 10M computations (210,7,100). CNN-kNN, PSC-kNN and RSP3-kNN were affected by the high level of noise that exists in this dataset. ENN-rule managed to remove many noisy items and consequently, CNN-rule, PSC and RSP3 achieved higher reduction rates when they executed on edited data.

Pendigits (Figure 4): FHACA had the best performance. It achieved accuracy values between 96.6%–98%. In all cases, it outperformed Hwang's algorithm. FHACA and RSP3 achieved higher accuracy than that of the conv-k-NN. The

most accurate (98%) FHACA classifier required 1.8M computations (50,10,100). A faster FHACA classifier required about 0.6M computations and achieved an accuracy of 97.26% (80,3,70).

Landsat satellite (Figure 5): Once again, FHACA performed very well. An accuracy of 90.75% was achieved with 533,282 computations (50,5,100). A FHACA classifier with half of that cost achieved an accuracy of over 90% (40,2,51/70/100). The highest FHACA accuracy was 90.9% and the corresponding cost was 1,067,000 computations (70,7,100). RSP3-kNN achieved an accuracy of 90.75, at a much higher cost. The other methods did not achieve accuracies above 90%.

Shuttle (Figure 6): Shuttle is an imbalanced dataset. Approximately 80% of the data belongs to one class. The two CNN approaches performed very well. This happened because the large class forms a "clear" and "tight" cluster and so, CNN-rule successfully removed a huge amount of data. The performance of RSP3 was close to that of CNN. Although FHACA was able to reach the accuracy level of the four reduction approaches, it required a higher cost. Hwang's algorithm achieved the highest accuracy value. Compared to the results of the previous four datasets, FHACA had worse perfomance. Although it achieved high accuracies (even 99,883%), it required high cost. This is because FHACA constructs many non-necessary representatives for the majority class. Nevertheless, FHACA performed comparably to the other methods when it classified test items belonging to rare classes.

Considering the experimental results on all datasets, we can conclude that the proposed model can achieve comparable or higher accuracy at a lower cost than the other methods.

5 Conclusion and Future work

We proposed a cluster-based model for speeding-up the k-NN classifier. The model involves the construction of a two level data structure and an algorithm that makes predictions using either the first or the second level of this data structure. Furthermore, the model lets the user define the desirable trade-off between accuracy and cost. Thus, it can be used either to improve the accuracy with gains in cost, or to significantly reduce the cost at the minimum level without sacrificing accuracy. Experimental results showed that significant performance improvement may be achieved, with the accuracy remaining at high levels (and comparable even to that of the conventional k-NN classifier).

We plan to incorporate in our method a mechanism for dynamically adapting the *NRRatio* parameter in relation to the number of representatives of each class. The main goal of this extension is to efficiently deal with imbalanced datasets. In addition, we will combine the proposed method with abstraction/filtering approaches and we will devise algorithms for the dynamic updating of SUDS.

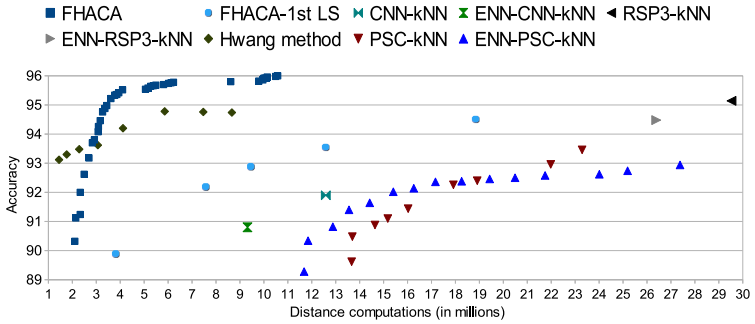


Fig. 2: Letter Recognition Dataset

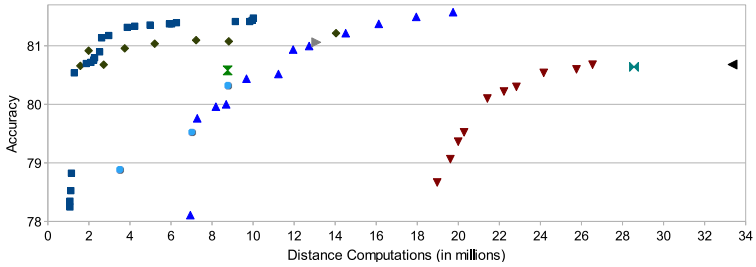


Fig. 3: Magic Gamma Telescope Dataset

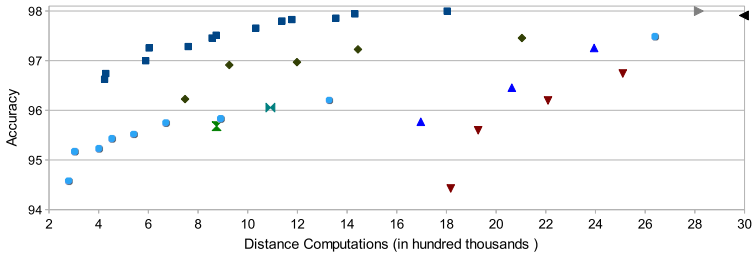


Fig. 4: Pendigits Dataset

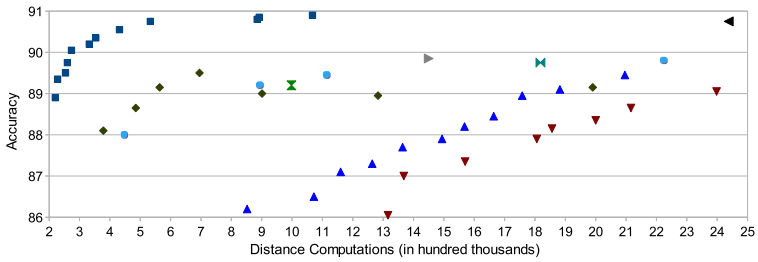


Fig. 5: Landsat Satellite Dataset

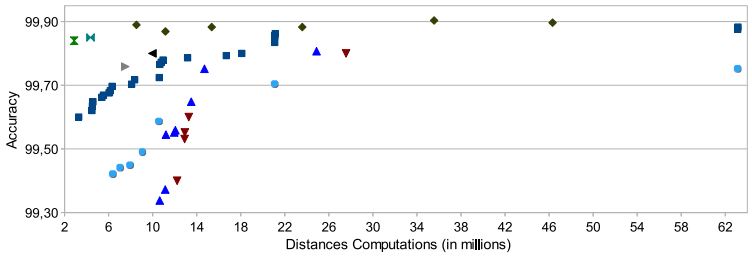


Fig. 6: Shuttle Dataset

References

1. 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)
2. Dasarathy, B.V.: Nearest neighbor (NN) norms : NN pattern classification techniques. IEEE Computer Society Press (1991)
3. Frank, A., Asuncion, A.: UCI machine learning repository (2010), `http://archive.ics.uci.edu/ml`
4. Garcia, S., Derrac, J., Cano, J., Herrera, F.: Prototype selection for nearest neighbor classification: Taxonomy and empirical study (in press). IEEE Transactions on Pattern Analysis and Machine Intelligence
5. Hart, P.E.: The condensed nearest neighbor rule. IEEE Transactions on Information Theory 14(3), 515–516 (1968)
6. 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)
7. Lozano, M.: Data Reduction Techniques in Classification processes (Phd Thesis). Universitat Jaume I (2007)
8. Mardia, K., Kent, J., Bibby, J.: Multivariate Analysis. Academic Press (1979)
9. 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)
10. Olvera-Lopez, J.A., Carrasco-Ochoa, J.A., Trinidad, J.F.M.: A new fast prototype selection method based on clustering. Pattern Anal. Appl. 13(2), 131–141 (2010)
11. Samet, H.: Foundations of multidimensional and metric data structures. The Morgan Kaufmann series in computer graphics, Elsevier/Morgan Kaufmann (2006)
12. Sánchez, J.S.: High training set size reduction by space partitioning and prototype abstraction. Pattern Recognition 37(7), 1561–1564 (2004)
13. Toussaint, G.: Proximity graphs for nearest neighbor decision rules: Recent progress. In: 34th Symposium on the INTERFACE. pp. 17–20 (2002)
14. Triguero, I., Derrac, J., Garcia, S., Herrera, F.: A taxonomy and experimental study on prototype generation for nearest neighbor classification (in press). IEEE Transactions on Systems Man and Cybernetics - Part C: Applications and Reviews
15. Wilson, D.R., Martinez, T.R.: Reduction techniques for instance-based learning algorithms. Machine Learning 38(3), 257–286 (2000)
16. Wilson, D.L.: Asymptotic properties of nearest neighbor rules using edited data. IEEE trans. on systems, man, and cybernetics 2(3), 408–421 (July 1972)
17. 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)