[9] H. Lin and L. Li, "Large-margin thresholded ensembles for ordinal regression: Theory and practice," in *Proc. 17th Int. Conf. Algorithmic Learn. Theory*, 2006, pp. 319–333.
[10] S. Kramer, G. Widmer, B. Pfahringer, and M. DeGroeve, "Prediction of ordinal classes using regression trees," *Fundamenta Informaticae*, vol. 47, pp. 1–13, 2001.
[11] S. Har-Peled, D. Roth, and D. Zimak, "Constraint classification: A new approach to multiclass classification and ranking," *Neural Inf. Process. Syst.*, vol. 15, pp. 785–792, 2002.
[12] R. Herbrich, T. Graepel, and K. Obermayer, "Lrage margin rank boundaries for ordinal regression," in *Advances in Large Margin Classifiers*. Cambridge, MA: MIT Press, 2000, pp. 115–132.
[13] P. McCullagh and J. A. Nelder, *Generalized Linear Models*. London, U.K.: Chapman & Hall, 1983.
[14] P. McCullagh, "Regression models for ordinal data," *J. Roy. Statist. Soc. B*, vol. 42, no. 2, pp. 109–142, 1980.
[15] V. E. Johnson and J. H. Albert, *Ordinal Data Modeling (Statistics for Social Science and Public Policy)*. New York: Springer-Verlag, 1999.
[16] A. Shashua and A. Levin, "Ranking with large margin principle: Two approaches," *Neural Inf. Process. Syst.*, vol. 15, pp. 937–944, 2002.
[17] H. Yu, J. Yang, and J. Han, "Classifying large data sets using SVMs with hierarchical clusters," in *Proc. 9th ACM SIGKDD Int. Conf. Knowl. Disc. Data Mining*, 2003, pp. 306–315.
[18] D. Boley and D. Cao, "Training support vector machine using adaptive clustering," in *Proc. 4th SIAM Int. Conf. Data Mining*, 2004, pp. 126–137.
[19] J. Wang, X. Wu, and C. Zhang, "Support vector machines based on k-means clustering for real-time business intelligence systems," *Int. J. Business Intell. Data Mining*, vol. 1, no. 1, pp. 54–64, 2005.
[20] J. Yuan, J. Li, and B. Zhang, "Learning concepts from large scale imbalanced data sets using support cluster machines," in *Proc. 14th Annu. ACM Int. Conf. Multimedia*, 2006, pp. 441–450.
[21] M. Almeida, A. Braga, and J. Braga, "SVM-KM: Speeding SVMs learning with a priori cluster selection and k-means," in *Proc. 6th Brazilian Symp. Neural Netw.*, 2000, pp. 162–167.
[22] H. Yu, J. Yang, J. Han, and X. Li, "Making svms scalable to large data sets using hierarchical cluster indexing," *Data Mining Knowl. Disc.*, vol. 11, no. 3, pp. 295–321, 2005.
[23] Z. Xu, K. Yu, V. Tresp, X. Xu, and J. Wang, "Representative sampling for text classification using support vector machines," in *Proc. 25th Eur. Conf. Inf. Retrieval Res.*, 2003, pp. 393–407.
[24] K. Zhang and J. T. Kwok, "Block-quantized kernel matrix for fast spectral embedding," in *Proc. 23rd Int. Conf. Mach. Learn.*, 2006, vol. 23, pp. 1097–1104.
[25] R. Bellman, *Introduction to Matrix Analysis*, 2nd ed. Philadelphia, PA: SIAM, 1997.
[26] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "An efficient k-means clustering algorithm: Analysis and implementation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 881–892, Jul. 2002.
[27] I. S. Dhillon, Y. Guan, and B. Kulis, "Weighted graph cuts without eigenvectors: A multilevel approach," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 11, pp. 1944–1957, Nov. 2007.
[28] W. Waegeman, B. Baets, and L. Boullart, "Roc analysis in ordinal regression learning," *Pattern Recognit. Lett.*, vol. 29, pp. 1–9, 2008.
[29] J. Rennie and N. Srebro, "Loss functions for preference levels: Regression with discrete, ordered labels," in *Proc. IJCAI Multidisciplinary Workshop Adv. Preference Handling*, 2005, pp. 180–186.

A Novel Template Reduction Approach for the K-Nearest Neighbor Method

Hatem A. Fayed and Amir F. Atiya

***Abstract*—The K-nearest neighbor (KNN) rule is one of the most widely used pattern classification algorithms. For large data sets, the computational demands for classifying patterns using KNN can be prohibitive. A way to alleviate this problem is through the condensing approach. This means we remove patterns that are more of a computational burden but do not contribute to better classification accuracy. In this brief, we propose a new condensing algorithm. The proposed idea is based on defining the so-called chain. This is a sequence of nearest neighbors from alternating classes. We make the point that patterns further down the chain are close to the classification boundary and based on that we set a cutoff for the patterns we keep in the training set. Experiments show that the proposed approach effectively reduces the number of prototypes while maintaining the same level of classification accuracy as the traditional KNN. Moreover, it is a simple and a fast condensing algorithm.**

***Index Terms*—Condensing, cross validation, editing, K-nearest neighbor (KNN), template reduction.**

I. Introduction

The K-nearest neighbor (KNN) classification rule is one of the most well-known and widely used nonparametric pattern classification methods. Its simplicity and effectiveness have led it to be widely used in a large number of classification problems, including handwritten digits, satellite image scenes, and medical diagnosis [1]–[5]. For KNN, however, two major outstanding problems are yet to be resolved by the research community. The first issue is the selection of the best K (number of neighbors to consider), as this problem is greatly affected by the finite sample nature of the problem. The second issue is the computational and the storage issue. The traditional KNN rule requires the storage of the whole training set which may be an excessive amount of storage for large data sets and leads to a large computation time in the classification stage. There are two well-known procedures for reducing the number of prototypes (sometimes referred to as template reduction techniques). The first approach, called "editing," processes the training set with the aim of increasing generalization capabilities. This is accomplished by removing prototypes that contribute to the misclassification rate, for example, removing "outlier" patterns or removing patterns that are surrounded mostly by others of different classes [6]–[8]. The second approach is called "condensing." The aim of this approach is to obtain a small template that is a subset of the training set without changing the nearest neighbor decision boundary substantially. The idea is that the patterns near the decision boundary are crucial to the KNN decision, but those far from the boundary do not affect the decision. Therefore, a systematic removal of these ineffective patterns helps to reduce the computation time. This can be established by reducing the number of prototypes that are centered in dense areas of the same class [9]–[20]. In this brief, we consider only

Manuscript received August 24, 2007; revised December 21, 2008; accepted March 11, 2009. First published April 21, 2009; current version published May 01, 2009.

H. A. Fayed is with the Department of Engineering Mathematics and Physics, Cairo University, Cairo 12613, Egypt (e-mail: h_fayed@eng.cu.edu.eg).

A. F. Atiya is with the Department of Computer Engineering, Cairo University, Cairo 12613, Egypt (e-mail: amir@alumni.caltech.edu).

Color versions of one or more of the figures in this brief are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNN.2009.2018547

the condensing approach. Below is a short summary of some existing algorithms for the condensing approach.

In 1968, Hart [9] was the first to propose an algorithm for reducing the size of the stored data for the nearest neighbor decision (the algorithm is called CNN). Hart defined a consistent subset of the data as one that classifies the remaining data correctly with the nearest neighbor rule. He built this consistent set by sequentially adding to it data points from the training set as long as the added data point is misclassified (using the 1-NN rule). By construction, the resulting reduced subset classifies all the training data correctly. Empirical results have shown that Hart's CNN rule considerably reduces the size of the training set at the expense of minimal or even no degradation in classification performance. The drawback of CNN is that frequently it may keep some points that are far from the decision boundary. To combat this, in 1972, Gates [10] proposed what he called the reduced nearest neighbor rule (RNN). This method is based on first applying CNN and then performing a postprocessing step. In this postprocessing step, the data points in the consistent set are revisited and removed if their deletion does not result in misclassifying any point in the training set. Experimental results confirmed that RNN yields a slightly smaller training subset than that obtained with CNN. In [11], Bhattacharya *et al.* (1992) proposed two methods, one based on the Voronoi graph and the other based on the Gabriel graph. The methods have the merit that they are exact and yield sets independent of the order in which the data are processed. The method based on a Voronoi graph yields a condensed set which is both training-set consistent (i.e., it classifies all the training data correctly) and decision-boundary consistent (i.e., it determines exactly the same decision boundary as that of the entire training set). However, it suffers from a large complexity due to the need to construct the Voronoi diagram. On the other hand, the method based on the Gabriel diagram is faster but it is neither decision-boundary consistent nor training-set consistent. In [12], Wilson and Martinez (2000) presented five algorithms for reducing the size of case bases: DROP1, DROP2, . . ., DROP5. Decremental reduction optimization procedure 1 (DROP1) is the basic removal scheme based on so-called associate patterns. The associate patterns for some pattern p are the patterns which have p as one of their K-nearest neighbors. The removal of p is determined based on its effect on the classification of its associates. DROP2 is a modification whereby the order of the patterns to be removed is selected according to a certain distance criterion in a way to remove patterns furthest from the decision boundary first. DROP2 also differs from DROP1 in that deletion decisions still rely on the original set of associates. DROP3, DROP4, and DROP5 are versions whereby noise-filtering pass is performed prior to applying the DROP2 procedure. In [13], Mollineda *et al.* (2002) obtained a condensed 1-NN classifier by merging the same class nearest clusters as long as the set of new representatives correctly classify all the original patterns. In [14], Wu *et al.* (2002) proposed an efficient method to reduce the training set required for KNN while maintaining the same level of classification accuracy, namely, the improved KNN (IKNN). This is implemented by iteratively eliminating patterns, which exhibit high attractive capacities (the attractive capacity s_y of a pattern y is defined as the number of patterns from class $C(y)$, which are closer to pattern y than other patterns belonging to other classes). The algorithm filters out a large portion of prototypes that are unlikely to match against the unknown pattern. This accelerates the classification procedure considerably, especially in cases where the dimensionality of the feature space is high. Other approaches for condensing are based on: 1) evolutionary algorithms and decision trees [15], 2) space partitioning [16], 3) decision boundary preservation [17], 4) estimation of the distribution of representatives according to the information they convey [18], 5) a gradient-descent technique for learning prototype positions and local metric weights [19], and 6) incorporation of both the proximity between patterns and geometrical distribution around the given pattern [20]. There are other methods that combine both editing and condensing techniques forming a hybrid model [21].

In this brief, we introduce a new condensing algorithm, namely, the template reduction for KNN (TRKNN). The basic idea is to define a "chain" of nearest neighbors. By setting a cutoff value for the distances among the chain, we effectively separate the patterns into the selected "condensed set" (probably consisting of patterns near the decision boundary) and the removed set (probably interior patterns). The paper is organized as follows. The proposed TRKNN method is described in Section II. Some analytical insights are introduced in Section III. Then the proposed method is validated experimentally in Section IV. Results are discussed in Section V. Finally, conclusions are drawn in Section VI.

II. Template Reduction for KNN

The goal of the proposed approach is to discard prototypes that are far from the boundaries and have little influence on the KNN classification. To establish this, we first introduce the so-called nearest neighbor chain. This is simply a sequence of the nearest neighbors from alternating classes. Consider first the following definitions. Consider a pattern $\mathbf{x}_i$ (also call it $\mathbf{x}_{i0}$ and let it be from class ω_m). Let $\mathbf{x}_{i1} \equiv \mathrm{NN}(\mathbf{x}_{i0})$ denote the nearest neighbor to $\mathbf{x}_{i0}$ that is from a different class. Similarly, let $\mathbf{x}_{i2} \equiv \mathrm{NN}(\mathbf{x}_{i1})$ denote the nearest neighbor to $\mathbf{x}_{i1}$ that is from the starting class (ω_m). We continue in this manner, with $\mathbf{x}_{i,j+1} \equiv \mathrm{NN}(\mathbf{x}_{ij})$. This sequence of $\mathbf{x}_{ij}$'s (whose class memberships alternates between class ω_m and the other classes) constitutes the nearest neighbor chain. Below is the precise definition.

Definition: A nearest neighbor chain C_i of a pattern $\mathbf{x}_i$ (of Class ω_m) is defined as the sequence $\mathbf{x}_{i0}, \mathbf{x}_{i1}, \mathbf{x}_{i2}, \ldots, \mathbf{x}_{ik}$ and the sequence $d_{i0}, d_{i1}, \ldots, d_{i,k-1}$ where the root pattern $\mathbf{x}_{i0} = \mathbf{x}_i$, and $\mathbf{x}_{ij}$ is the closest pattern to $\mathbf{x}_{i,j-1}$ (of a class different from ω_m if j is odd and of Class ω_m if j is even). Moreover, $d_{ij} = \|\mathbf{x}_{i,j+1} - \mathbf{x}_{ij}\|_2$ is the Euclidean distance between patterns $\mathbf{x}_{i,j+1}$ and $\mathbf{x}_{ij}$. The chain is stopped at $\mathbf{x}_{ik}$ if $\mathbf{x}_{i,k+1} = \mathbf{x}_{i,k-1}$. Note that the distance sequence is a nonincreasing sequence (i.e., $d_{ij} \geq d_{i,j+1}$).

Fig. 1 shows some examples of constructed chains for a two-class problem. In summary, a chain is constructed as follows. Start from a pattern $\mathbf{x}_i$. Find the nearest neighbor from a different class. Then, from that pattern find the nearest neighbor from the starting class. Continue in this manner until we end up with two patterns that are nearest neighbor to each other. Note that by construction the distances between the patterns in the chain form a nonincreasing sequence. Note also that patterns downstream the chain will probably be close to the classification boundary, because they will have smaller distances from the patterns of different classes. This provides the basis of the proposed condensing procedure.

The basic idea of the proposed condensing approach is as follows. For each pattern $\mathbf{x}_i$ in the training set, we construct its corresponding chain C_i. The pattern $\mathbf{x}_{ij}$ in the chain is dropped (from the selected condensed set) if $d_{ij} > \alpha \cdot d_{i,j+1}$ where α is a threshold >1, and $j = 0, 2, 4, \ldots$ up to the size of the chain. Note that we allow only patterns from the same class as that of $\mathbf{x}_i$ to be eliminated (i.e., we consider only the even patterns in the chain). This is important when dealing with a multiclass problem as the chain is constructed using the *one-against-all* concept as has been illustrated earlier. Typically, when starting the chain with an interior point, the distance to the next point in the chain will be large. As the chain converges onto the boundary points, the distances decrease in value and will more or less level off. This gives a rationale for the proposed cutoff procedure. Because there is a significant decrease in distances, the considered pattern is deemed to be probably an interior point and can be discarded, whereas if the

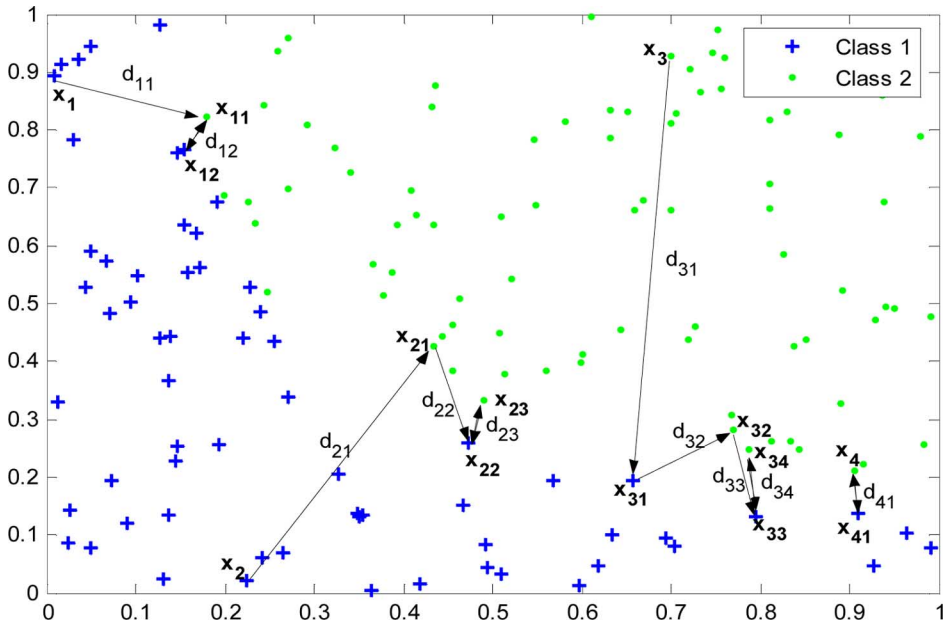

Fig. 1. Illustrative example of the chains.

distances do not decrease too much, then we are probably oscillating around the classification boundary, and the pattern is kept. Below are the precise steps of the condensing method.

Algorithm: TRKNN

Inputs:

- Training set Ω.
- Distance ratio threshold α.

Output:

- Reduced training set Ω.

Method:

For each pattern $\mathbf{x}_i$ in Ω

Find its corresponding chain C_i

End For

For each chain C_i

For $j = 0, 2, 4, \ldots$ up to the size of C_i

If $d_{ij} > \alpha \cdot d_{i,j+1}$ then mark the pattern $\mathbf{x}_{ij}$

End For

End For

Drop all marked patterns from Ω.

Fig. 1 shows an example that illustrates the working of the algorithm. The closest pattern of different class to pattern $\mathbf{x}_1$ is $\mathbf{x}_{11}$ and the distance between them is d_{11}. Similarly, the closest pattern of different class to pattern $\mathbf{x}_{11}$ is $\mathbf{x}_{12}$ and the distance between them is d_{12}. Now $\mathbf{x}_1$ is dropped if $d_{11} > \alpha \cdot d_{12}$.

Note that some computational savings can be achieved by observing that some chains encompass some other smaller chains. By operating on the larger chains first, we automatically get the condensing results of the smaller chains contained in them, leading to some computational shortcut. Note also that the condensed set does not depend on some pattern presentation order, like many condensing algorithms. The reason is that at each step the full training set is considered as a whole. Another comment that applies also to most condensing methods is that after condensing it is imperative to have the number of neighbors K (for the KNN classification) a little reduced than K used with the whole data set. This is due to the somewhat redundant patterns being removed. This reduction will be bigger if more data is removed.

TABLE I
SUMMARY OF DATA SETS

Data set	Number of patterns	Number of features	Number of classes
Breast Cancer	699	9	2
Pima Indians	768	8	2
Balance scale	625	4	3
Landsat	6435	36	6
Pendigits	10992	16	10

III. ANALYTICAL INSIGHTS

The proposed algorithm relies on the concept that a pattern near the classification boundary will tend to have a nearest neighbor (from another class) that is fairly close. On the other hand for an interior pattern that distance will tend to be larger. Based on this concept, we discard the latter patterns. While this concept is fairly intuitive, it would be beneficial to provide some analytical analysis in order to gain insight and to understand the degree and the factors affecting that relationship. Below we develop some approximate analysis.

For simplicity, consider a two-class case. Consider a pattern $\mathbf{x}_0$ from class ω_1, and let the dimension of the pattern vector be D. Moreover, let $p(\mathbf{x}|\omega_2)$ denote the class conditional density for class ω_2, and let there be N training patterns from class ω_2. Finally, let $\mathrm{NN}(\mathbf{x}_0)$ denote the nearest neighbor to pattern $\mathbf{x}_0$ that is from class ω_2. Let r be the distance from $\mathbf{x}_0$ to $\mathrm{NN}(\mathbf{x}_0)$. Define the following random variable:

$$v = \int_{\|\mathbf{x}-\mathbf{x}_0\|_2 \le r} p(\mathbf{x}|\omega_2)dx.$$

TABLE II
SUMMARY OF AVERAGE TRAINING TIME AND TEST TIME IN SECONDS

Data set	KNN		DROP2		IKNN		TRKNN	
	Train	Test	Train	Test	Train	Test	Train	Test
Breast Cancer	0.06	0.06	0.56	0.03	0.20	0.03	0.06	0.03
Pima Indians	0.05	0.08	0.21	0.05	0.12	0.08	0.07	0.08
Balance scale	0.05	0.07	0.49	0.06	0.11	0.06	0.06	0.06
Landsat	1.34	6.29	10.15	5.86	10.13	5.83	3.90	4.02
Pendigits	3.14	15.61	25.93	7.90	15.91	7.16	6.30	6.83

TABLE III
MEAN NUMBER OF PROTOTYPES (NPROT) AND MEAN TEST ERROR RATES (ERR) OVER THE DIFFERENT FOLDS. STANDARD DEVIATIONS ARE SHOWN IN BRACKETS

Data set	KNN		DROP2		IKNN		TRKNN	
	NPROT	ERR (%)	NPROT	ERR (%)	NPROT	ERR (%)	NPROT	ERR (%)
Breast Cancer	350	3.72 (0.87)	82 (9.4)	3.66 (1.08)	128 (16.4)	3.78 (0.92)	91 (29.4)	3.95 (0.98)
Pima Indians	384	27.68 (2.80)	134 (13.4)	33.85 (2.87)	291 (8.8)	27.68 (2.80)	211 (21.1)	27.86 (1.71)
Balance scale	313	13.18 (1.29)	145 (10.9)	14.43 (1.38)	187 (6.3)	13.89 (1.98)	155 (9.8)	14.98 (1.99)
Landsat	3218	10.77 (0.53)	2006 (90.7)	11.46 (0.80)	1924 (139.1)	10.73 (0.33)	1872 (66.7)	11.03 (0.32)
Pendigits	5496	0.86 (0.11)	3932 (45.4)	0.91 (0.10)	3189 (247.1)	0.87 (0.11)	2994 (186.0)	0.97 (0.10)

It is well known (see [22] and [23]) that such a variable obeys the following beta density function:

$$p(v) = N(1-v)^{N-1}, \qquad 0 \le v \le 1.$$

Assume that the number of training patterns from class ω_2 is sufficiently large such that the $\mathrm{NN}(\mathbf{x}_0)$ is close to $\mathbf{x}_0$. Hence, within the ball centered around $\mathbf{x}_0$ stretching out to $\mathrm{NN}(\mathbf{x}_0)$, the density $p(\mathbf{x}|\omega_2)$ can be approximated as almost constant, and hence

$$v \approx c \cdot r^D p(\mathbf{x}_0|\omega_2)$$

where the term $c \cdot r^D$ represents the volume of a D-dimensional ball of radius r, with $c \equiv \pi^{D/2}/\Gamma(1+D/2)$. Then, we get

$$r \approx \left(\frac{v}{c \cdot p(\mathbf{x}_0|\omega_2)} \right)^{1/D}.$$

The expectation of r is then given by

$$E(r) = \int_0^1 N(1-v)^{N-1} \left(\frac{v}{c \cdot p(\mathbf{x}_0|\omega_2)} \right)^{1/D} dv$$

which can be evaluated as

$$E(r) = \frac{\Gamma(N+1)\Gamma(1+1/D)(\Gamma(1+D/2))^{1/D}}{\sqrt{\pi}\Gamma(N+1+1/D)} p(\mathbf{x}_0|\omega_2)^{-1/D}.$$

The previous relation confirms the fact that the distance to the nearest neighbor from the other class is small if we are close to the classification boundary (where the opposing class-conditional density $p(\mathbf{x}|\omega_2)$ would be high). Conversely, that distance would be large if $p(\mathbf{x}|\omega_2)$ was small (signifying that we are in the interior and far away from the boundary). Moreover, that monotone relationship decays more slowly for large dimensional problems. One might contemplate the situation when we are near the boundary but $p(\mathbf{x}|\omega_2)$ is still small. This situation arises when the other class conditional density $p(\mathbf{x}|\omega_1)$ is also small, that is, we operate in a relatively sparse area in the space. To compensate for that, the algorithm uses the other distances in the chain to have a "relative" cutoff point. That is, we discard patterns based on comparing them according to the successive distances in the chain (i.e., when $d_{ij} > \alpha \cdot d_{i,j+1}$).

IV. EXPERIMENTAL RESULTS

To validate the proposed algorithm, we compared it with the traditional KNN and some of the condensing methods: the DROP2 [12] and the IKNN [14] for several real-world data sets. Note that, as we focus on comparing condensing methods, we do not employ DROP3, DROP4, or DROP5 [12], as these just add preprocessing steps that can be applied to any method. On the other hand, DROP1's accuracy is significantly low compared to KNN and DROP2. Therefore, to attain a fair comparison, DROP2 is included in the comparison. We use the fivefold validation procedure for the purpose of tuning the key parameters of each method. In this approach, the training set is partitioned into five equal parts. Training is performed on four parts and validated on the fifth part. Then the validation part is rotated and training is performed again. The process is repeated five times, and the validation classification error on all five parts is combined. The parameter values that yield the best validation results will then be chosen for testing the performance. The tuned parameters are as follows. In all methods, suggested values for K are odd values from 1 to 9. For IKNN, suggested values for the attractive capacity threshold (S) are: $[0.01, 0.05, 0.1] \times N_{\min}$, where $N_{\min}$ is the minimum number of patterns corresponding to the same class while suggested values for the portion function are: $\zeta(t) = \beta \times (t+1)^{-0.5}$, where $\beta \in \{0.1, 0.2\}$ (for more details and description of the parameters, see [14]). For TRKNN, suggested values for α are 1.2, 1.4, and 1.6. The distance metric used is the Euclidean distance for all methods. Concerning the DROP2 method, there are no tunable parameters. There are four main performance measures. The training time represents the time it takes to condense the training set, including searching for the optimal parameters, such as K and the others. (However, it is computed as the average time per tested parameter set. This way we avoid penalizing methods that have a larger number of parameters, such as the competing IKNN, or have finer parameter grid.) The testing time

TABLE IV
COMBINED 5 × 2 CROSS-VALIDATION F TEST FOR THE TEST CLASSIFICATION ERROR (ERR) AND THE NUMBER OF PROTOTYPES (NPROT). REJECTION DECISION OF THE NULL HYPOTHESIS IS SHOWN IN BRACKETS

Data set		Breast Cancer	Pima Indians	Balance Scale	Landsat	Pendigits
TRKNN vs. KNN	NPROT	70.68 **(Yes)**	125.34 **(Yes)**	292.06 **(Yes)**	270.74 **(Yes)**	581.29 **(Yes)**
	ERR	0.65 (No)	0.61 (No)	1.32 (No)	1.85 (No)	2.75 (No)
TRKNN vs. DROP2	NPROT	0.74 (No)	9.48 **(Yes)**	1.09 (No)	3.31 (No)	116.5 **(Yes)**
	ERR	1.25 (No)	4.96 **(Yes)**	0.62 (No)	3.24 (No)	3.19 (No)
TRKNN vs. IKNN	NPROT	2.18 (No)	28.61 **(Yes)**	54.16 **(Yes)**	2.04 (No)	1.81 (No)
	ERR	1.00 (No)	0.64 (No)	1.09 (No)	4.47 (No)	1.33 (No)

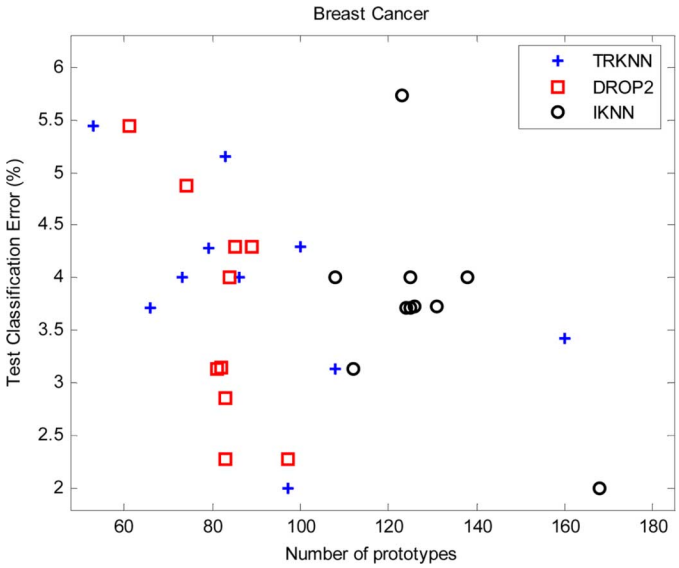

Fig. 2. Test classification error versus the number of prototypes over the 5 × 2 folds for the Breast Cancer data set.

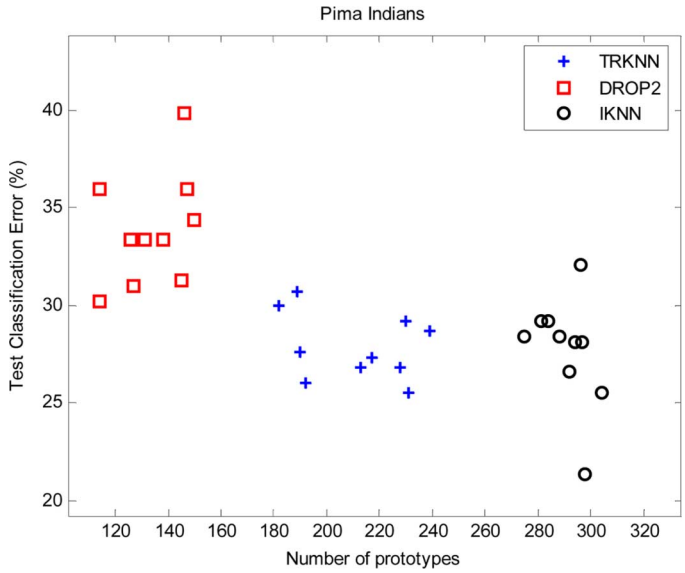

Fig. 3. Test classification error versus the number of prototypes over the 5 × 2 folds for the Pima Indians data set.

represents the classification time using the condensed set. It is very closely tied to the third measure, which is the number of patterns in the condensed set. The test classification error is the final performance measure. The main goal of any condensing method is to reduce the number of patterns as much as possible, with as little sacrifice as possible to the classification accuracy.

In comparing the classification error and the number of prototypes of any two methods a challenging issue is to test whether the difference is statistically significant. We have used the combined 5 × 2 cross-validation F test [24] (a study in [25] shows the superiority of this tests compared to alternative ones). To apply this test, five replications of twofold cross validation are performed. In each replication, the data set is divided into two equal-sized sets, one for training and the other for testing. Let $p_i^{(j)}$ be the difference between the error rates of the two classifiers on fold $j = 1, 2$ for replication $i = 1, \ldots, 5$. Let the average on replication i be $\bar{p} = (p_i^{(1)} + p_i^{(2)})/2$ and let the estimated variance be $s_i^2 = (p_i^{(1)} - \bar{p})^2 + (p_i^{(2)} - \bar{p})^2$. The combined 5 × 2 cross-validation F test is applied by assuming that the statistic $f = \sum_{i=1}^{5}\sum_{j=1}^{2}(p_i^{(j)})^2 / 2\sum_{i=1}^{5} s_i^2$ has approximately an F distribution with ten and five degrees of freedom. The null hypothesis is rejected with 95% confidence if f is greater than 4.74.

We used five real-world data sets. The first data set was obtained from cancer1.dt file from Proben1 database [26], which was created based on the "Breast Cancer Wisconsin" problem data set from the University of California at Irvine (UCI) Machine Learning Repository database [27]. The second data set was obtained from diabetes1.dt file from Proben1 database [26], which was created based on the "Pima Indians Diabetes" problem data set from the UCI Machine Learning Repository database [27]. The remaining data sets were obtained from the UCI repository [27]. Summary of the data sets is shown in Table I. It shows the names and the details of the data sets, such as the number

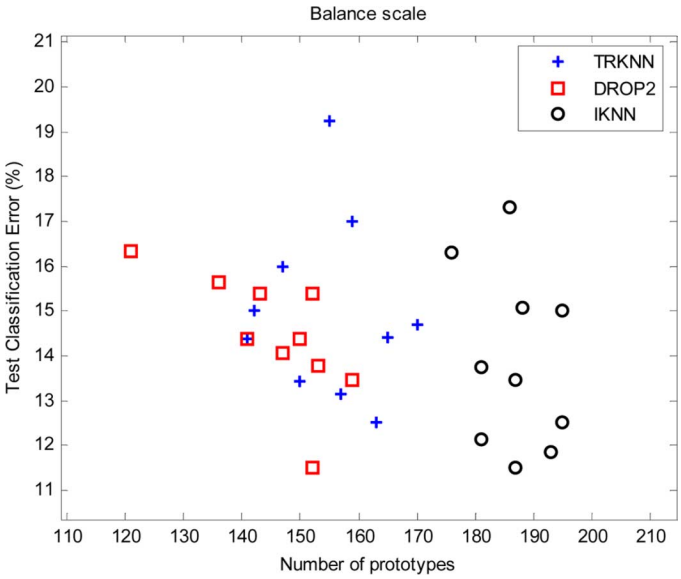

Fig. 4. Test classification error versus the number of prototypes over the 5 × 2 folds for the Balance Scale data set.

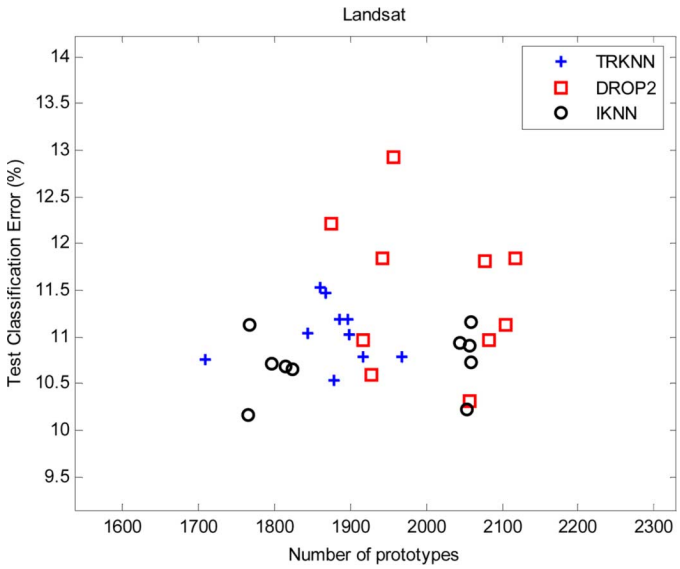

Fig. 5. Test classification error versus the number of prototypes over the 5 × 2 folds for the Landsat data set.

of patterns, the number of features, and the number of classes. We performed the implementation using MATLAB 7 on Windows XP with SP2 running on Pentium IV 2.4-GHz PC with 768-MB RAM.

V. Results

The results of the average training time over all folds and the test time (in seconds) are shown in Table II while the number of prototypes and the test classification error rates are shown in Table III. The results of the 5 × 2 significance test applied to the test classification error and to the number of prototypes are shown in Table IV for the different data sets. The table shows that for all data sets the F test at the 95% confidence level does not reject the hypothesis that IKNN and TRKNN give a similar test error, and also does not reject the hypothesis that KNN and TRKNN give a similar test error. On the other hand, TRKNN gives smaller number of prototypes (i.e., higher reduction rate) than IKNN for two of the data sets (Pima Indians and Balance Scale), also at the 95% level. For the three other data sets, TRKNN's outperformance is not significant at the 95% level. Compared to DROP2, we observe the following. For the Pima Indians data set, DROP2 gets significantly lower number of prototypes. But this is at the expense of significantly worse test error (as if DROP2 dropped too many patterns to an extent that it affected classification performance). On the other hand, for the Pendigits data set, TRKNN produces a significantly lower number of prototypes, while test error is comparable.

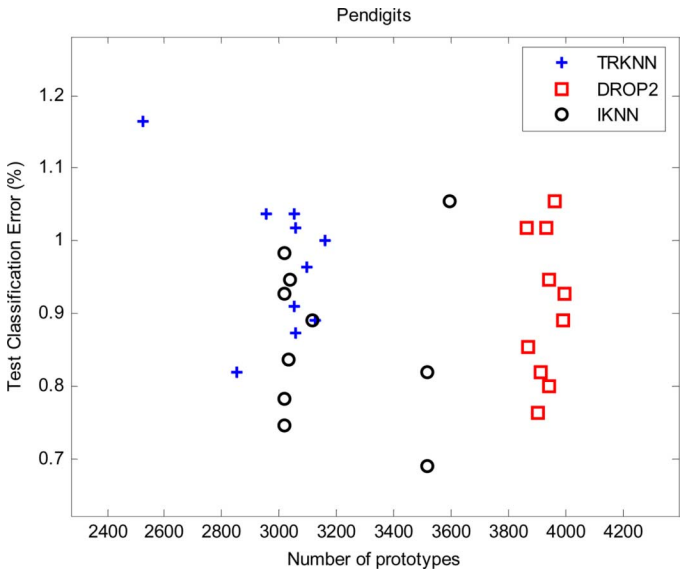

Fig. 6. Test classification error versus the number of prototypes over the 5 × 2 folds for the Pendigits data set.

We note that there is generally a tradeoff between test error and number of prototypes (NPROT) selected (or reduction ratio). To clarify that the proposed TRKNN is winning in the overall NPROT/test error tradeoff, we performed the following analysis. It is based on the performed 5 × 2 test. As mentioned, we perform the test ten times (on the two folds times the five partitions). Consider a single test (i.e., on one specific fold and one specific partition) and record NPROT and the test error. Repeat ten times for the ten tests and get ten pairs of NPROT/test error numbers. Plot these pairs as points in the 2-D space with the axes being NPROT and the test error. We will have ten points for TRKNN, corresponding to the NPROT/test error outcomes of the ten test sets, ten other points for IKNN, corresponding also to the test outcomes for IKNN, and ten other points for DROP2. Figs. 2–6 show the plots for each of the tested UCI data sets. One can see in the plots that for all problems the test errors for both TRKNN and IKNN are comparable. On the other hand, for three of the problems (Breast Cancer, Pima Indians, and Balance Scale) the number of prototypes for TRKNN is significantly lower than that of IKNN. For one problem (Pendigits), TRKNN beats in the NPROT aspect (i.e., gives lower NPROT), but by a small amount. For the remaining problem (Landsat), both methods are about equal. When we say "significantly beats" it is based on the fact that the averages are different and the standard deviations do not lead to overlapping of the points, which can be seen visually in the plot. For some of the problems (such as Balance Scale), IKNN gives lower average test error, but there is a large overlap of the points (in the test error dimension), and that makes the difference not statistically significant. Concerning DROP2 versus TRKNN, as observed before, one can see that DROP2 obtains lower NPROT at the expense of a worse test error for Pima Indians data set. On the other hand, for both Landsat and Pendigits, DROP2 produces significantly worse (i.e., higher) NPROT.

This happens while the test errors for DROP2 and TRKNN are comparable. One can see the clear separation in the NPROT dimension, while in the test error dimension the data overlap. It seems that possibly for larger data sets DROP2 does not prune out enough points (which paradoxically are the type of problems where we need to drop points the most).

Concerning the training time (Table II), the time of TRKNN is considerably shorter than that of IKNN, by a factor of around 2 or 3 times. This is because the training time of TRKNN is dominated by the computation of the distance matrix (the matrix that holds distances between pairs of all training patterns) whose complexity is $O(n^2)$, where n is the training set size. This distance matrix is computed only once at the beginning of the training process. For IKNN, besides the computation of the distance matrix, there is an extra computation at each iteration for the evaluation and sorting of the attractive capacities with complexity $O(nA \log A)$, where A is the average attractive capacity. For large data sets, this attractive capacity A could be rather large. This computation is repeated for a number of iterations (say J), leading to the extra complexity of $O(nJA \log A)$ (beyond that of TRKNN). Similarly, TRKNN is faster (in training speed) than DROP2, by a factor of around 3 or 4. The reason for the slow training speed for DROP2 is the need to sort the distances that are computed from each pattern to its "enemy pattern." The enemy pattern is defined as the closest pattern from a different class.

Overall, viewing all performance criteria, such as the test error/number of prototype tradeoff, and the speed, we feel that TRKNN has an edge over the competing IKNN and DROP2 methods.

VI. Conclusion

In this brief, a new condensing method for KNN is proposed. The method drops patterns that are far away from the boundary and thus have little influence on the KNN classification. Experiments show that the proposed approach reduces the template set size without sacrificing the accuracy compared to the traditional KNN and two recent condensing methods. In addition, this method can be considered simple in implementation, and is computationally fast.

References

[1] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, 2nd ed. New York: Academic, 1990.

[2] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. New York: Wiley, 2001.

[3] T. Hastie, R. Tibshirani, and J. H. Friedman, *The Elements of Statistical Learning: Data Mining Inference, and Prediction*, ser. Statistics. Berlin, Germany: Springer-Verlag, 2001.

[4] A. R. Webb, *Statistical Pattern Recognition*, 2nd ed. London, U.K.: Wiley, 2002.

[5] W. Duch, "Similarity based methods: A general framework for classification, approximation and association," *Control Cybern.*, vol. 29, no. 4, pp. 937–968, 2000.

[6] D. L. Wilson, "Asymptotic properties of nearest neighbor rules using edited data," *IEEE Trans. Syst. Man Cybern.*, vol. SMC-2, no. 3, pp. 408–420, Jul. 1972.

[7] P. A. Devijver and J. Kittler, "On the edited nearest neighbor rule," in *Proc. 5th Int. Conf. Pattern Recognit.*, Miami, FL, 1980, pp. 72–80.

[8] F. J. Ferri and E. Vidal, "Colour image segmentation and labeling through multiedit-condensing," *Pattern Recognit. Lett.*, vol. 13, pp. 561–568, 1992.

[9] P. E. Hart, "The condensed nearest neighbor rule," *IEEE Trans. Inf. Theory*, vol. IT-14, no. 3, pp. 515–516, May 1968.

[10] W. Gates, "The reduced nearest neighbor rule," *IEEE Trans. Inf. Theory*, vol. IT-18, no. 3, pp. 431–433, May 1972.

[11] B. K. Bhattacharya, R. S. Poulsen, and G. T. Toussaint, "Application of proximity graphs to editing nearest neighbor decision rules," in *Proc. 16th Symp. Interface Between Comput. Sci. Statist.*, 1984, pp. 97–108.

[12] D. R. Wilson and T. R. Martinez, "Reduction techniques for instance-based learning algorithms," *Mach. Learn.*, vol. 38, no. 3, pp. 257–286, 2000.

[13] R. A. Mollineda, F. J. Ferri, and E. Vidal, "An efficient prototype merging strategy for the condensed 1-NN rule through class-conditional hierarchical clustering," *Pattern Recognit.*, vol. 35, pp. 2771–2782, 2002.

[14] Y. Wu, K. Ianakiev, and V. Govindaraju, "Improved k-nearest neighbor classification," *Pattern Recognit.*, vol. 35, pp. 2311–2318, 2002.

[15] J. R. Cano, F. Herrera, and M. Lozano, "Using evolutionary algorithms as instance selection for data reduction in KDD: An experimental study," *IEEE Trans. Evol. Comput.*, vol. 7, no. 6, pp. 561–575, Dec. 2003.

[16] J. S. Sánchez, "High training set size reduction by space partitioning and prototype abstraction," *Pattern Recognit.*, vol. 37, no. 7, pp. 1561–1564, 2004.

[17] R. Barandela, F. J. Ferri, and J. S. Sánchez, "Decision boundary preserving prototype selection for nearest neighbor classification," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 19, no. 6, pp. 787–806, 2005.

[18] D. Huang and T. W. S. Chow, "Enhancing density-based data reduction using entropy," *Neural Comput.*, vol. 18, no. 2, pp. 470–495, 2006.

[19] R. Paredes and E. Vidal, "Learning prototypes and distances: A prototype reduction technique based on nearest neighbor error minimization," *Pattern Recognit.*, vol. 39, no. 2, pp. 171–179, 2006.

[20] J. S. Sánchez and A. I. Marqués, "An LVQ-based adaptive algorithm for learning from very small codebooks," *Neurocomputing*, vol. 69, no. 7–9, pp. 922–927, 2006.

[21] H. Brighton and C. Mellish, "Advances in instance selection for instance-based learning algorithms," *Data Mining Knowl. Disc.*, vol. 6, pp. 153–172, 2002.

[22] D. A. Fraser, *Nonparametric Methods in Statistics*. New York: Wiley, 1957, ch. 4.

[23] R. D. Short and K. Fukunaga, "The optimal distance measure for nearest neighbor classification," *IEEE Trans. Inf. Theory*, vol. IT-27, no. 5, pp. 622–627, Sep. 1981.

[24] E. Alpaydin, "Combined 5×2 cv F test for comparing supervised classification learning algorithms," *Neural Comput.*, vol. 11, pp. 1885–1892, 1999.

[25] T. G. Dietterch, "Approximate statistical tests for comparing supervised classification learning algorithms," *Neural Comput.*, vol. 10, pp. 1895–1923, 1998.

[26] L. Prechelt, "Proben1, A set of neural-network benchmark problems," University of Karlsruhe, Germany, 1994 [Online]. Available: Available: http://page.mi.fu-berlin.de/prechelt/Biblio/1994-21.pdf

[27] C. L. Blake and C. J. Merz, "UCI Repository of Machine Learning database," Dept. Inf. Comput. Sci., Univ. California, Irvine, Irvine, CA, 1998 [Online]. Available: http://www.ics.uci.edu/~mlearn