A Proposal for Local k Values for k-Nearest Neighbor Rule

Nicolás García-Pedrajas, Juan A. Romero del Castillo, and Gonzalo Cerruela-García

***Abstract*— The k-nearest neighbor (k-NN) classifier is one of the most widely used methods of classification due to several interesting features, including good generalization and easy implementation. Although simple, it is usually able to match and even outperform more sophisticated and complex methods. One of the problems with this approach is fixing the appropriate value of k. Although a good value might be obtained using cross validation, it is unlikely that the same value could be optimal for the whole space spanned by the training set. It is evident that different regions of the feature space would require different values of k due to the different distributions of prototypes. The situation of a query instance in the center of a class is very different from the situation of a query instance near the boundary between two classes. In this brief, we present a simple yet powerful approach to setting a local value of k. We associate a potentially different k to every prototype and obtain the best value of k by optimizing a criterion consisting of the local and global effects of the different k values in the neighborhood of the prototype. The proposed method has a fast training stage and the same complexity as the standard k-NN approach at the testing stage. The experiments show that this simple approach can significantly outperform the standard k-NN rule for both standard and class-imbalanced problems in a large set of different problems.**

***Index Terms*— Classification, class-imbalanced data sets, k-nearest neighbors (k-NN).**

I. Introduction

A classification problem of K classes and N training observations consists of a set of instances whose class membership is known. Let $T = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$ be a set of N training samples where each instance $\mathbf{x}_j$ belongs to a domain X. Each label is an integer from the set $Y = \{1, \ldots, K\}$. A classifier is a function $f : X \to Y$ that maps an instance $\mathbf{x} \in X \subset \mathbb{R}^D$ to an element of Y. The task is to find a definition for the unknown function $\mathscr{F}(\mathbf{x})$, given the set of training instances.

The k-nearest neighbor (k-NN) rule is a well-known and widely used method of classification. The method consists of storing a set of prototypes that represent the knowledge of the problem. The k-NN method is used mainly because of its simplicity and its ability to achieve error results comparable to much more complex methods. For instance, in computer vision, it has been applied successfully to a wide range of problems, such as face recognition [1], articulated pose estimation [2], and character recognition [3].

The k-NN method, although simple, can usually match and even outperform more sophisticated and complex methods in terms of generalization error. One of the problems with this classifier, however, is fixing the appropriate value of k. Although a good value might be obtained using cross validation (CV), the same value is unlikely to be optimal for the whole space spanned by the training set. In this brief, we propose a simple method to use and train local values for the k parameter. Our approach is based on learning the local value of k directly from the training set, evaluating the effect of every value of k and choosing the best performing one. The proposed method is fast and accurate, showing better generalization capabilities than the original method with similar complexity.

Manuscript received January 13, 2015; revised September 14, 2015 and December 2, 2015; accepted December 6, 2015. Date of publication January 29, 2016; date of current version January 17, 2017. This work was supported by the Spanish Ministry of Science and Innovation through the Junta de Andalucía under Project P07-TIC-02682, Project P09-TIC-4623, and Project TIN-2011-22967.

The authors are with the Department of Computing and Numerical Analysis, University of Córdoba, Córdoba 14004, Spain (e-mail: npedrajas@uco.es; aromero@uco.es; gcerruela@uco.es).

Digital Object Identifier 10.1109/TNNLS.2015.2506821

The selection of the optimal value of k in a certain data set is always a problem, as only a finite amount of training data is available. The standard approach assumes that there exists a unique k value that is optimal for all the regions of the input space. In practice, the situations of different prototypes differ. For instance, the appropriate values of k are very different for a prototype in the middle of a class, near the boundaries of the class or with many noisy instances in its neighborhood. Thus, the use of different k values for the different prototypes might have a positive effect on the classification accuracy of the k-NN rule.

Our approach is based on two hypotheses. First, we assign to each prototype a value of k, whose ideal value would be the optimal number of neighbors to be used in its neighborhood. Second, to obtain the value associated with each prototype, we consider local performance values. Thus, instead of the standard set formed by prototypes of the form $(\mathbf{x}, y_i)$, where $\mathbf{x}_i$ is the prototype and y_i is its category, we use augmented prototypes of the form $(\mathbf{x}, y_i, k_i)$, where k_i is the associated k value for prototype $\mathbf{x}_i$ that will be used in its neighborhood as the value for the k-NN rule.

Our approach has the advantage of allowing the selection of a local k value with a very simple and fast procedure. In training time, the method is simple and has linear complexity; in testing time, the algorithm has the same workload of the standard version of the k-NN rule. Furthermore, the whole process can be run in parallel, which means that the size of the data set is not a constraint for our approach, as the process of obtaining the optimal k value for each prototype is independent of the remaining prototypes.

This brief is organized as follows. Section II shows some related work; Section III explains our proposal; Section IV describes our experimental setup; Section V shows the results of our experiments; and finally, Section VI states the conclusion of this brief and describes some new research lines.

II. Related Work

Although a few methods have been proposed for the different types of nonglobal k values, none of those methods have shown significant improvement over the standard approach of a global k value set by CV. For the nearest neighbor rule, Gao *et al.* [4] developed a two-level method. At the low level, they used Euclidean distance to determine a local subspace centered at an unlabeled test sample. At the high level, AdaBoost was used as guidance for local information extraction. These works and many others are focused on learning local metrics, but very few have tried to obtain the local values of k.

Ferrer-Troyano *et al.* [5] proposed the k-frequent nearest neighbor algorithm. In this algorithm, two limit values for k are obtained using the prototypes. To classify a new example q, the k-NN algorithm is applied several times to the same example q by varying the value of k within the limits obtained. This method is an attempt to remove

Algorithm 1 Outline of the Proposed Algorithm

Data : A training set $T = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, $\mathbf{x}_i \in \mathbb{R}^D$, a minimum value of k, k_{min}, and a maximum value of k, k_{max}.

Result : The vector of local k values $\mathbf{k}$.

Obtain vector of 10-fold cross-validation accuracy for global values of k from k_{min} to k_{max}.

for *every* $x \in T$ **do**

  for $k = k_{min}$ *to* $k = k_{max}$ **do**

    Obtain $eval(k)$ using eq. 1

  end

  Obtain optimal local value of k

  Assign optimal value of k to k_i

end

Return $\mathbf{k}$

the need to use a fixed value of k for all the instances. However, there is no local value of k for each instance. Wang *et al.* [6] developed a method for the local estimation of k called confident-nearest neighbor method. However, the experimental results showed that this method performed worse than the standard k-NN rule.

Wettschereck and Dietterich [7] developed a model that stores, for each sample, a list of k values that correctly classified the instance using a leave-one-out method. To classify a query instance q, M-nearest neighbors are obtained, and the k value that correctly classifies most of these M neighbors is chosen. However, this method and a few variances also proposed in the paper did not show any improvement over the standard k-NN approach.

III. Optimally Local k Value for k-NN Rule

Our approach is based on assigning a local k value for the neighborhood of each prototype. The testing process is as follows. For a query instance of unknown category, the nearest neighbor in the training set is obtained. Then, the associated k value of that nearest neighbor is used to classify the query instance. The training process must obtain the k value that will be associated with every prototype. To obtain the best k associated with each prototype, we propose a greedy approach. For each prototype, we test the performance of all k values in an interval $[k_{\min}, k_{\max}]$ given by the user and choose the value that shows the best performance. The process is shown in Algorithm 1.

One of the advantages of this approach is that to obtain the local k value associated with a prototype $\mathbf{x}_i$, we only need to consider the instances, whose nearest neighbor is $\mathbf{x}_i$. Therefore, the evaluation of all the values in the interval $[k_{\min}, k_{\max}]$ is very fast. However, it might also be a problem, as for many instances, the number of nearest neighbors may be small or even zero. To avoid the negative effect of considering too few neighbors, to obtain the local value of k for every instance, we use the n-nearest neighbors instead of only just the nearest one. This has the effect of smoothing the estimation of the best k value. In our experiments, we used the value of $n = 3$. The results are similar with other small values.

In evaluating each k value, our first objective is the classification performance, which is measured as the classification accuracy for standard problems and the geometric mean of specificity and sensitivity (see Section IV) for class-imbalanced problems. In case of a tie, where different values of k have the same evaluation, we arbitrarily choose the smallest one. As stated above, the only instances considered are the instances for which one of the n-nearest neighbors is the current prototype.

However, for many instances, there are many values of k with the same performance value. For example, there may be many values of k for which all the neighbors of an instance are correctly classified. In such a case, we do not know which value is optimal. To address that situation, we add to the local performance measure of each value of k, a global performance measure. For each value of k, we obtain the global classification performance, $\mathrm{acc}_{\mathrm{global}}(k)$, using tenfold CV.[1] Then, the evaluation of each k value, $\mathrm{eval}(k)$, is the combination of the local accuracy of k, $\mathrm{acc}_{\mathrm{local}}(k)$, and the global accuracy, $\mathrm{acc}_{\mathrm{global}}(k)$

$$\mathrm{eval}(k) = \mathrm{acc}_{\mathrm{local}}(k) + \mathrm{acc}_{\mathrm{global}}(k). \quad (1)$$

The effect of this combination of local and global measures is twofold. First, it serves as a way to break ties when many values of k have the same performance. Second, it adds a global view that acts as a smoothing factor to avoid excessively large local variations of k.

The proposed method has a fast-training stage. To estimate the optimal k values for each prototype, we only evaluate a few of its neighbors, three in our experiments. In testing time, our proposal has the same complexity as the standard k-NN approach.

IV. Experimental Setup

Our approach can be used with almost any version of k-NN. Thus, we considered different k-NN rules, such as the standard k-NN, the edited k-NN [8], the adaptive k-NN [9], and the symmetrical k-NN [10]. The major difference among the adaptive, standard, and symmetrical k-NN rules is the distance measure used in each one. With these four different versions of k-NN rule, we show the wide applicability of our method. In the experiments, a wide interval of k was used, setting $k_{\min} = 1$ and $k_{\max} = 100$. For all the four standard k-NN rules, the value of k was chosen by means of tenfold CV in the same interval $k \in [1, 100]$. Results for edited k-NN are not reported as they were always significantly worse than the other three methods. However, our approach also achieved significantly better results that the standard version for the edited k-NN. $\mathrm{acc}_{\mathrm{local}}(k)$, see Algorithm 1, was obtained using leave-one-out estimation method. In the following, the standard, the adaptive, and the symmetrical k-NN rules will be named Rule 1, Rule 2, and Rule 3, respectively. The standard way of obtaining the optimal value of k will be named standard training method (STM) and our local approach proposed training method (PTM).

To make a fair comparison between standard algorithms and our proposed approach, we have selected a set of 80 problems for standard data sets and 65 problems for class-imbalanced data sets. The data sets for standard problems are from the Univesity of California at Irvine (UCI) machine learning repository [11], the Kent ridge bio-medical data set (http://datam.i2r.a-star.edu.sg/datasets/krbd/index.html), and our own data sets. These data sets are a good example of a wide range of number of instances, from 148 to 20 000, and of number of features, from 4 to 20 000. To estimate the classifier performance, we used tenfold CV.

For the class-imbalanced data sets, we used a set of 65 problems that were taken from the UCI Machine Learning Repository, from the Kent Ridge Bio-medical Data set, and from our own data sets. These data sets are a good example of a wide range of imbalance ratios, from 1:2 to 1:3841; of number of instances, from 57 to 1 700 517; and of number of features, from 3 to 23 549. Thus, they form a challenging set of problems for any method.

The source code, written in C and licensed under the GNU General Public License, used for all methods, as well

[1]That means that although fast our algorithm is always slower than the tenfold CV method.

as the partitions of the data sets, is freely available at http://cib.uco.es/index.php/supplementary-material-for-localk or upon request from us. We used the Wilcoxon test as the main statistical test for comparing the pairs of algorithms with a significance level of 0.05.

For the standard data sets, we used the standard measure of accuracy, namely the percentage of instances correctly classified. However, recent works have shown that misclassification rates may be biased because they contain substantial randomness [12]. Thus, as an additional measure and an alternative to the misclassification rate, we have also used Cohen's κ measure, which is a method that compensates for random hits. Its original purpose was to measure the degree of agreement. However, κ can also be adapted to measure the classification accuracy, and its use is recommended, because it takes random successes into consideration [12]. The value of κ can be computed from the confusion matrix in a classification task as follows:

$$\kappa = \frac{n \sum_{i=1}^{C} x_{ii} - \sum_{i=1}^{C} x_{i.}x_{.i}}{n^2 - \sum_{i=1}^{C} x_{i.}x_{.i}} \tag{2}$$

where x_{ii} is the cell count on the main diagonal, n is the number of examples, C is the number of classes, and $x_{i.}$ and $x_{.i}$ are the column and row total counts, respectively. The value of κ ranges from -1 (total disagreement) to 1 (perfect agreement). For multiclass problems, κ is a very useful yet simple metric for measuring the accuracy of the classifier, while compensating for random successes.

Accuracy is not a useful measure for imbalanced data. Given the number of true positives, false positives, true negatives, and false negatives, we can define several measures. Two of the most common are the sensitivity (Sn) and the specificity (Sp). From these basic measures, others have been proposed, such as the $G-$mean measure: $G - \text{mean} = \sqrt{Sp \cdot Sn}$. In additional, the area under the receiver-operating characteristic curve (auROC) is a useful metric for classifier performance, because it is independent of the decision criterion selected and prior probabilities. Thus, we will use auROC as the main measure of classifier performance for class-imbalanced data sets.

V. Experimental Results

In this section, we show results and perform several statistical tests for standard and class-imbalanced data sets.

A. Standard Data Sets

Our experiments were aimed at comparing the performance of our proposal against the standard k-NN classifier. Using the data sets shown in Section IV, we compared the performance of the three different k-NN rules using our approach. As we are comparing our approach with the standard method, we use a Wilcoxon test [13]. Table I shows the comparison of our method for the different versions of k-NN rule in terms of accuracy and κ measure. Table I shows the win/loss record of the algorithms and the p-values, and R^+/R^- of the Wilcoxon test for the results over the 80 data sets. As a general rule, we consider a confidence level of 95%.

Table I shows that our approach was able to improve the standard method for all the three different rules. Wilcoxon test finds significant differences for all cases and both the measures. The results are shown in Figs. 1 and 2, for accuracy and κ measure, respectively, where each point represents the result for a data set of both the methods. Points over the main diagonal represent data sets for which our approach achieved a better performance.

It is interesting to study the behavior of our approach and the standard methods. Fig. 3 shows the average values of k for our approach and the optimal value of k found by tenfold CV in logarithmic scale. Each point represents both the values for the same data set. Fig. 3 shows that when CV found small values of k, our approach found somewhat larger values. However, when CV obtained larger values, our algorithm tended to find smaller ones. It seems that our proposal tended to avoid extreme k values.

TABLE I

Comparison of Our Approach for the Three Versions of k-NN Rule in Terms of Accuracy and κ Measure

Accuracy	Rule 1		Rule 2		Rule 3	
	STM	PTM	STM	PTM	STM	PTM
Average	0.8281	0.8320	0.8021	0.8042	0.8293	0.8335
Win/loss		49/22		44/21		50/23
Wilcoxon		0.0034		0.0031		0.0017
R^+/R^-		2231.5/1008.5		2237.0/1003.0		2273.5/966.5

κ measure	k-NN		Rule 2		Rule 3	
	STM	PTM	STM	PTM	STM	PTM
Average	0.6333	0.6472	0.6015	0.6060	0.6352	0.6474
Win/loss		54/20		42/26		51/27
Wilcoxon		0.0000		0.0192		0.0006
R^+/R^-		2507.5/732.5		2108.0/1132.0		2337.5/902.5

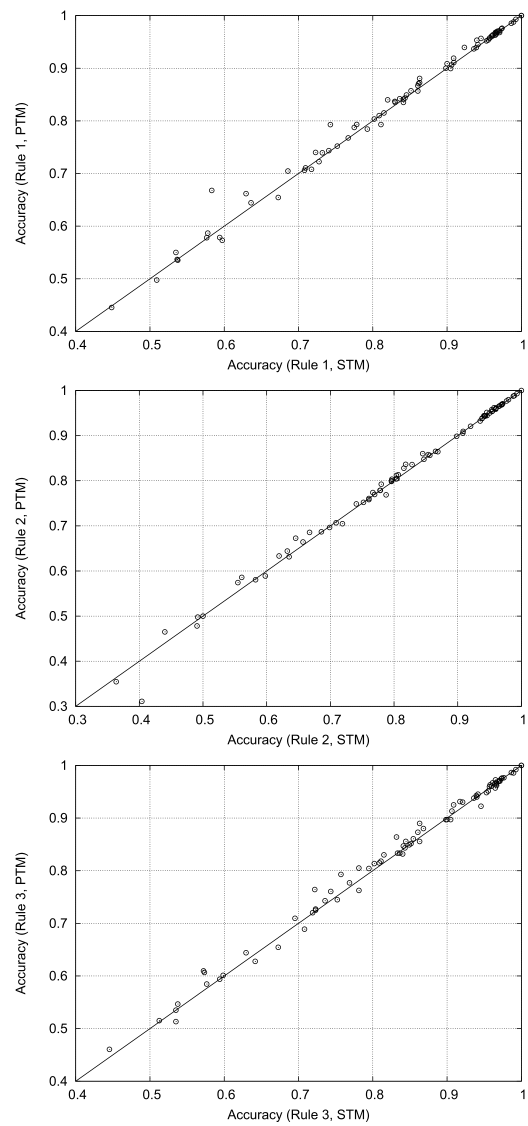

Fig. 1. Results of both methods in terms of accuracy for every data set for the three k-NN rules.

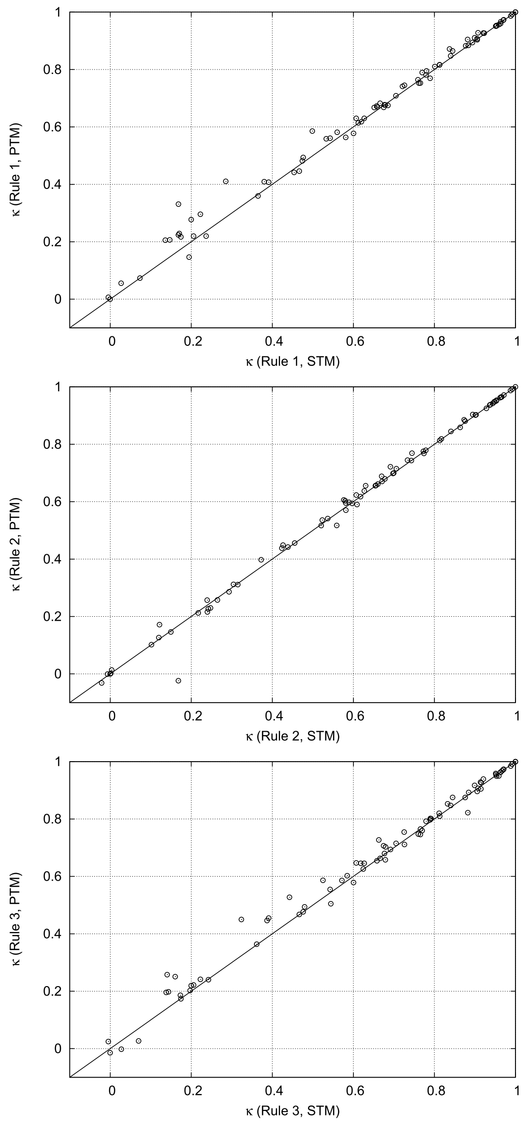

Fig. 2. Results of both methods in terms of κ measure for every data set for the three k-NN rules.

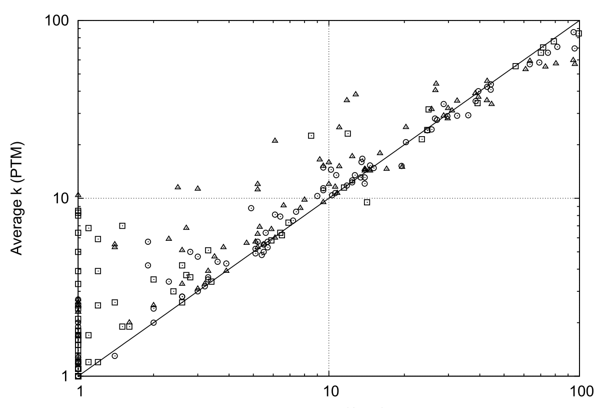

Fig. 3. Average k values for STM and PTM for Rule 1 (circles), Rule 2 (squares), and Rule 3 (triangles).

B. Class-Imbalanced Data Sets

The second step was to test our approach in class-imbalanced data sets. First, we applied undersampling to account for the class-imbalance problem. A comparison in terms of G-mean measure and auROC is shown in Table II for the results over the 65 data sets. Wilcoxon test finds the differences significant for all the three methods and both the measures.

Figs. 4 and 5 represent these results for G-mean and auROC, respectively. Each point represents the result for a data set of both the methods. Points over the main diagonal represent data sets for which our approach achieved a better performance.

TABLE II

COMPARISON OF OUR APPROACH FOR THE THREE VERSIONS OF k-NN RULES IN TERMS OF G-MEAN AND auROC

	G-mean					
	Rule 1		Rule 2		Rule 3	
	STM	PTM	STM	PTM	STM	PTM
Average	0.7680	0.8162	0.7811	0.8023	0.7755	0.8287
Win/loss		50/15		37/23		51/14
Wilcoxon		0.0000		0.0214		0.0000
R^+ / R^-		1909.0/ 236.0		1424.5/ 720.5		1915.0/ 230.0

	auROC					
	Rule 1		Rule 2		Rule 3	
	STM	PTM	STM	PTM	STM	PTM
Average	0.8539	0.8743	0.8376	0.8417	0.8649	0.8810
Win/loss		39/26		43/18		37/28
Wilcoxon		0.0084		0.0198		0.0166
R^+ / R^-		1476.0/ 669.0		1429.0/ 716.0		1439.0/ 706.0

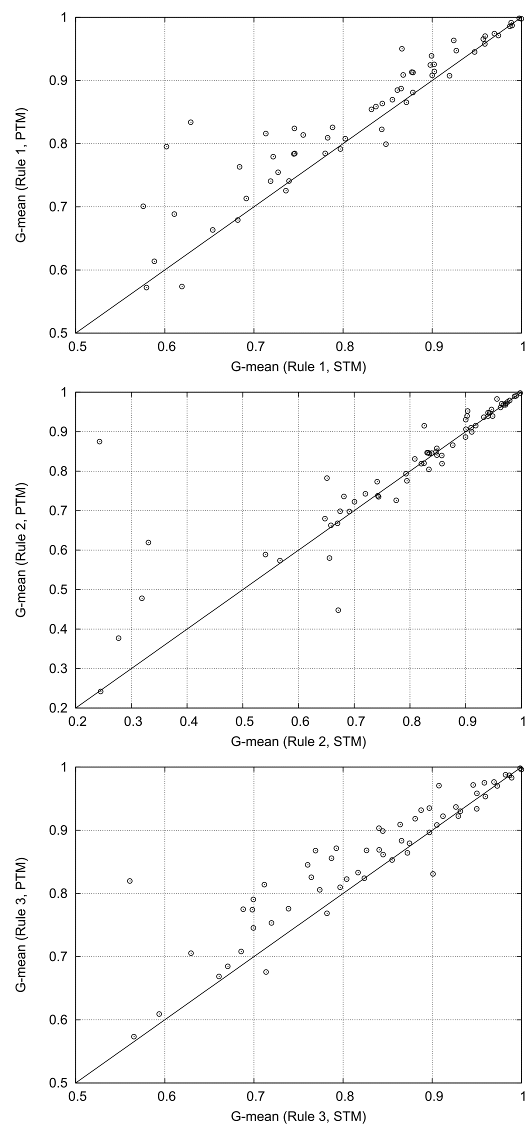

Fig. 4. Results of both methods for every class-imbalanced data set in terms of G-mean for the three k-NN rules.

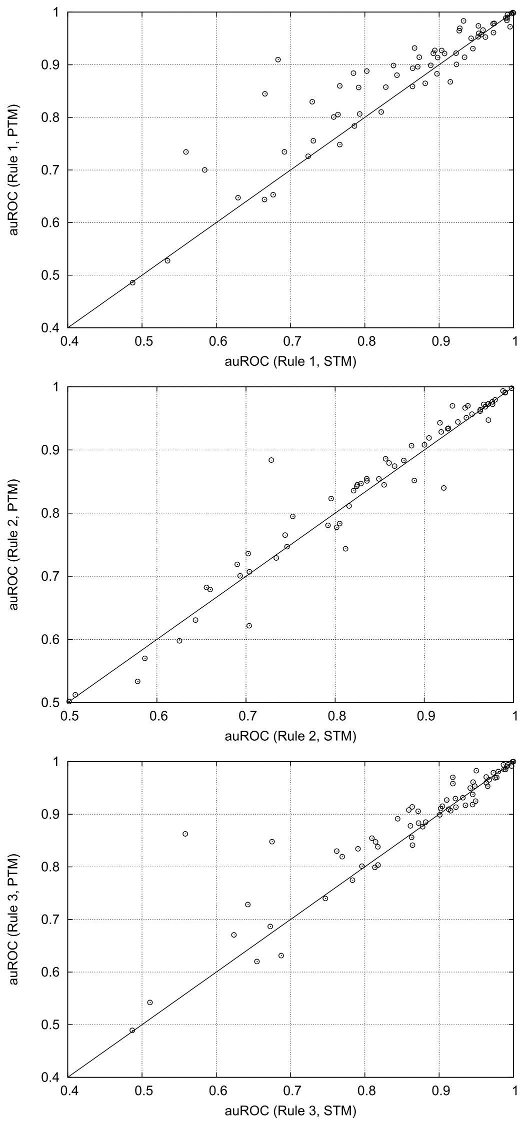

Fig. 5. Results of both methods for every class-imbalanced data set in terms of auROC for the three k-NN rules.

Fig. 6 shows the average k values obtained by the standard tenfold CV method and our approach. It is interesting to note that for class-imbalanced data sets, the differences between the values of k obtained by our proposal and tenfold CV were larger than for the case of standard data sets. A possible explanation is that for class-imbalanced data sets, the local differences are more marked due to the uneven distribution of samples in the classes, and thus there is more variation in the values of k across the space. Furthermore, for the class-imbalanced data sets, the value obtained by CV was larger in most cases for the standard approach. A possible explanation is in the global nature of the standard tenfold CV approach for obtaining k. For regions where there are almost no instances of the minority class, which are fairly common, especially for highly imbalanced data sets, there are many values of k for which there is no performance difference as the estimated training error in that area is always zero. Thus, when larger values of k are needed in other regions, those values are always globally preferred. However, when we use our method, this issue does not occur, as we can select an independent local value.

Finally, we have stated that our approach is fast. Table III shows the minimum, average, and maximum training times for our approach

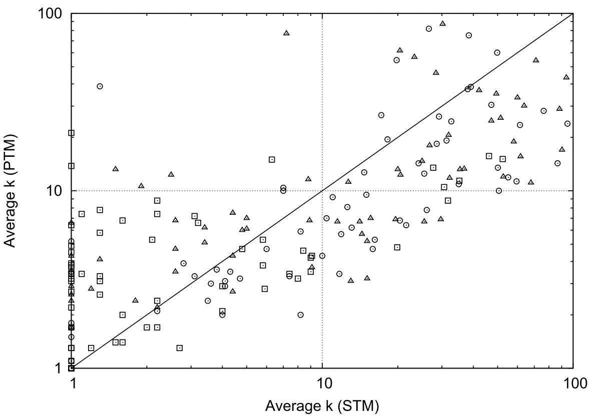

Fig. 6. Average k values for STM and PTM for class-imbalanced data sets for Rule 1 (circles), Rule 2 (squares), and Rule 3 (triangles).

TABLE III

MINIMUM, AVERAGE, AND MAXIMUM TRAINING TIMES IN SECONDS FOR PTM FOR STANDARD AND CLASS-IMBALANCED PROBLEMS

	Standard datasets			Class-imbalanced datasets		
	Minimum	Average	Maximum	Minimum	Average	Maximum
Rule 1	0.1	114.4	1814.2	0.0	33.9	819.8
Rule 2	0.2	50.7	726.0	0.0	29.7	770.6
Rule 3	0.6	28316.6	522552.0	0.0	5677.8	146943.0

for all the methods. STM is not showed, as it has no training step. Table III shows reasonably fast times for the three rules. Rule 3 is clearly slower, however, the training times are still within feasible bounds and also a parallel implementation would significantly reduce them.

VI. CONCLUSION AND FUTURE WORK

In this brief, we have shown a method for introducing a local value of k for the k-NN classifier that has demonstrated a better performance than the standard k-NN for both the balanced and imbalanced data sets. The method consists of associating a local value of k with every prototype and using that k value for the nearest neighbor of the prototype. We presented a method of obtaining those local values of k using a fast algorithm based on the local effect of each value.

The method is compared with the standard approach of obtaining the best value of k using tenfold CV and the four different versions of k-NN rule. The proposed method demonstrated a better performance for both the standard and class-imbalanced data sets, using two different classification performance measures in both the cases. We have also studied the average values for k obtained by our approach and compared them with the standard CV method.

The time needed to obtain the optimal k value for each prototype has been shown to be small, and furthermore, the whole process can be run in parallel, which means that the size of the data set is not a constraint for our approach.

Many different lines of research may be opened from this approach. The most obvious means of improving this brief is to couple the local value of k with instance selection. An evolutionary algorithm performing both the tasks simultaneously would be a very promising approach. In such an evolutionary method, the value of k for each instance and whether an instance is removed would be simultaneously considered. Furthermore, the same philosophy presented here can be applied to the other forms of k-NNs, such as the fuzzy k-NN and distance-weighted k-NN algorithms.

References

[1] L. Wiskott, J.-M. Fellous, N. Krüger, and C. von der Malsburg, "Face recognition by elastic bunch graph matching," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 7, pp. 775–779, Jul. 1997.

[2] G. Shakhnarovich, P. Viola, and T. Darrell, "Fast pose estimation with parameter-sensitive hashing," in *Proc. 9th IEEE Int. Conf. Comput. Vis.*, Nice, France, Oct. 2003, pp. 750–757.

[3] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 4, pp. 509–522, Apr. 2002.

[4] Y. Gao, J. Pan, G. Ji, and Z. Yang, "A novel two-level nearest neighbor classification algorithm using an adaptive distance metric," *Knowl.-Based Syst.*, vol. 26, pp. 103–110, Feb. 2012.

[5] F. J. Ferrer-Troyano, J. S. Aguilar-Ruiz, and J. C. Riquelme, "Non-parametric nearest neighbor with local adaptation," in *Proc. 10th Portuguese Conf. Artif. Intell.*, vol. 2258. 2001, pp. 22–29.

[6] J. Wang, P. Neskovic, and L. N. Cooper, "Neighborhood size selection in the k-nearest-neighbor rule using statistical confidence," *Pattern Recognit.*, vol. 39, no. 3, pp. 417–423, 2006.

[7] D. Wettschereck and T. G. Dietterich, "Locally adaptive nearest neighbor algorithms," in *Advances in Neural Information Processing Systems*, vol. 6. San Mateo, CA, USA: Morgan Kaufmann, 1994, pp. 184–191.

[8] D. L. Wilson, "Asymptotic properties of nearest neighbor rules using edited data," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-2, no. 3, pp. 408–421, Jul. 1972.

[9] J. Wang, P. Neskovic, and L. N. Cooper, "Improving nearest neighbor rule with a simple adaptive distance measure," *Pattern Recognit. Lett.*, vol. 28, no. 2, pp. 207–213, 2007.

[10] R. Nock, M. Sebban, and D. Bernard, "A simple locally adaptive nearest neighbor rule with application to pollution forecasting," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 17, no. 8, pp. 1369–1382, 2003.

[11] M. Lichman, "UCI machine learning repository," School Inf. Comput. Sci., Univ. California, Irvine, CA, USA, 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[12] A. Ben-David, "A lot of randomness is hiding accuracy," *Eng. Appl. Artif. Intell.*, vol. 20, no. 7, pp. 875–885, 2007.

[13] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, Dec. 2006.