NEAREST NEIGHBOR DECISION RULE FOR PIXEL CLASSIFICATION IN REMOTE SENSING

SZYMON GRABOWSKI[1], ADAM JÓŹWIK[1,2], C.H. CHEN[3]

[1]*Technical University of Łódź, Computer Engineering Department*
Al. Politechniki 11, 90-924 Łódź, E-mail: SGrabow@zly.kis.p.lodz.pl

[2]*Institute Biocybernetics and Biomedical Engineering, Polish Academy of Sciences*
02-109 Warsaw, Trojdena 4

[3]*University of Massachusetts Dartmouth, Electrical and Computer Engineering Dept., 285 Old Westport Road, N.Dartmouth*

A classifier based on the *k*-NN rule is known as the one that offers a very good performance. Learning of such kind of classifiers consists in determination the value of *k*. Some modification of the standard *k*-NN rule may lead to the improvement of the classification quality. The relatively new *k* Nearest Centroid Neighbor (*k*-NCN) decision rule uses an interesting concept of surrounding neighborhood, that is such a neighborhood, which takes into account not only the proximity of neighbors, but also their spatial location. Neighbors should be located not only close to a query sample, but also possibly around it in the space. In this chapter we present our decision rule called *k* Near Surrounding Neighbors (*k*-NSN), which "improves" the neighborhood used in *k*-NCN with respect to both described aspects. Moreover, we present a voting technique which finds several *k* parameters for *k*-NN, *k*-NCN and *k*-NSN rules learnt from random partitions of the training set and utilizes them in an ensemble of classifiers. As opposed to most ensemble methods, our algorithms require moderate computational increase in relation to the base classifiers, and even almost negligible computation increase in the voting *k*-NN case. We test the aforementioned methods on a remote sensing dataset (already used in several experiments) and obtain results which show attractiveness of the presented concepts in applications where prediction accuracy is of primal importance. The main disadvantage of the *k*-NN decision rule and its modified versions is a necessity of keeping the whole training set, as the reference set, in the computer memory during a classification phase. Numerous procedures, which have been already proposed for reference set reduction, concern the 1-NN rule. Although most proposed methods were originally devised for the 1-NN rule, there is no obstruction to use the received reduced sets with *k*-NN classifiers. It is also possible to reclassify the original reference set by applying the *k*-NN rule, standard or modified, and then to use the simple 1-NN rule with the reclassified set. The effectiveness of these approaches will be studied in relation to four different algorithms of reference set size reduction.

1 Introduction

Remote sensing image analysis, in its final stage, consists in classification of pixels. The construction of a classifier is based on the large training set. Furthermore, not only a good performance but also the speed of classification phase plays a very important role in a choice of the classifier type. The best possible decision rule offers the Bayes classifier that operates according to the formula: $p(j/\mathbf{x})=p(j)\cdot f(\mathbf{x}/j)/f(\mathbf{x})$, where $p(j/\mathbf{x})$ is a probability of the class j under assumption that the classified object is described by a feature vector $\mathbf{x}$, $f(\mathbf{x}/j)$ denotes the density of probability distribution for the class j and $f(\mathbf{x})$ is the density of probability distribution of the feature vector $\mathbf{x}$. The vector $\mathbf{x}$ is assigned to the class j that corresponds to the maximum value of $p(j/\mathbf{x})$. For convenience, the feature vectors $\mathbf{x}$ will be treated also as points in the feature space.

All functions which appear on the right side in the above mentioned Bayes formula are unknown. To approximate them one can use the neighborhood containing the k nearest neighbors of the classified point $\mathbf{x}$. In this way, $p(j)\approx m_j/m$, $f(\mathbf{x}/j)\approx k_j/(m_j\cdot V(\mathbf{x},k))$, $f(\mathbf{x})\approx k/(m\cdot V(\mathbf{x},k))$, where m_j is a number of objects from the class j in the training set, m is a numerical force of the training set, k_j means a number of points from the class j among k nearest points (neighbors) of the classified point $\mathbf{x}$ and $V(\mathbf{x},k)$ is a volume occupied by a

hypersphere containing these k nearest neighbors. Thus, the left side can be calculated. The probability function that occurs on the left side of the Bayes formula is then approximated by the ratio k_j/k, i.e. $p(j/\mathbf{x}) \approx k_j/k$. In this way the k nearest neighbor (k-NN) decision rule has been obtained. The classified point $\mathbf{x}$ is assigned to the class j that corresponds to the highest value of k_j/k. It is proved by others authors [1] that if the size m of the training set gets larger ($m \to \infty$), $k \to \infty$ and $k/m \to 0$ then the performance of the k-NN rule converges to the performance of Bayes' classifier. That is why the classifier based on k-NN rule has been chosen as a subject of the present paper. The k-NN rule was originally proposed in [2].

The training set containing points with known class membership is the set that is used for the classifier construction. In the case of the k-NN rule, it may serve for experimental determination a value of the parameter k. For instance, we can used the well known *leave-one-out* misclassification rate estimation or a cross-validation technique [3] to select the value of k which offers the best performance.

In recent years the concept of so-called *surrounding neighborhood* has been introduced. Such neighborhood can intuitively be understood as an item subject to two complementary constraints. Firstly, the neighbors of a query point q should be as **close** to it as possible. Secondly, the neighbors should also be located as **symmetrically** around q as possible. The k-NN ignores the latter aspect. Let us concentrate on one practical proposal fitting into this framework.

The idea of Chaudhuri [4], later developed to a decision rule [5], seems to be an interesting attempt to focus on neighbors, which are located not only close enough to the given sample, but also possibly homogeneously distributed around the sample. It is the concept of Nearest Centroid Neighborhood (NCN). The k nearest centroid neighbors of a query point q are obtained as follows:

– first neighbor of q is its nearest neighbor, n_1;

– the i'th neighbor, n_i, $i \geq 2$, is such that the centroid (i.e. the mean) c_i of this and previously selected neighbors, n_1, ..., n_{i-1}, is the closest to q.

Because of the centroid criterion, the spatial distribution of neighbors is taken into account. On the other hand, the incremental nature of the way in which successive neighbors are obtained guarantees their proximity to the query sample q.

Experiments conducted by Sánchez et al. ([5, 6][1]) confirm attractiveness of the k-NCN decision rule, especially in applications where classification accuracy is more important than classification time. The k-NCN usually outperformed the standard k-NN rule.

Just like for k-NN, also for k-NCN the number of neighbors must be estimated with respect to the training set, preferably with the leave-one-out method.

In [7] we proposed another surrounding neighborhood based decision rule, called k Near Surrounding Neighborhood (k-NSN), which tries to optimize both criteria used by k-NCN.

In this chapter we propose an ensemble of k-NN (or k-NCN, or k-NSN…) classifiers in which the values of k are diversified via estimations performed on various random partitions of the training set. We believe such an approach is, to a certain degree, a protection from overfitting. An experimental confirmation of the idea is presented in

[1] In our experiments on the remote sensing dataset used in the paper and on several UCI datasets, the better of the two modifications to the k-NCN rule proposed in [6], offered a notably worse accuracy than the original rule.

Section 5. Moreover, as opposed to most classification schemes with voting over multiple classifiers, the proposed technique does not require an increase in computational resources directly proportional to the number of components.

2 The *k* Near Surrounding Neighbors (*k*-NSN) decision rule

The heuristic nature of *k*-NCN encourages us to search for other decision rules which would take into account both described aspects of neighborhood. As the centroid criterion from *k*-NCN seems really good, we decided not to change it, but only optimize the set of neighbors according to both criteria: one related to proximity and the other to the distance of the neighbor set's centroid to the test sample.

Our classifier first searches for *k* NCN neighbors of a test sample and then in a loop tries to exchange some neighbors with other samples which are both closer and better in the sense of the centroid criterion to the test sample.

More precisely, the proposed decision rule, which we called *k Near Surrounding Neighbors* (*k*-NSN), operates as described in Fig. 1.

Note it is a *random mutation hill climbing* algorithm; the neighbor exchanging idea is analogous to the one applied by Skalak to prototype selection [8].

In our experiments the learning phase for *k*-NSN was performed with use of the *k*-NCN rule, which is faster.

Find k nearest centroid neighbors of a test sample q. Call the neighbors n_1, ..., n_k. Call their centroid c.
For *counter* = 1 to ITERATIONS do
{
  Select a random neighbor n_i, $1 \le i \le k$
  Select a random sample s, such that $d(q,s) \le d(q,n_{farthest})$, where $n_{farthest}$ is the k'th, according to the distance, NCN neighbor of q
  If $d(q,s) \le d(q,n_i)$
  {
    Let *tentative_c* = centroid(n_1, ..., n_{i-1}, s, n_{i+1}, ..., n_k)
    If $d(q,tentative\_c) < d(q,c)$
    {
      Let $n_i = s$
      Let c = *tentative_c*
    }
  }
}
Return the set of neighbors n_1, ..., n_k.

Figure 1: The *k* Near Surrounding Neighbors (*k*-NSN) decision rule

3 Voting over multiple classifiers

An essential problem in all classification tasks is the danger of overfitting. This phenomenon consists in choosing the classifier and its parameters (comprising e.g. some numeric values, its reference set(s) *etc*.) so well fitting the learning data that it does not actually fit the real concept. The trouble „as such" is inevitable since in practice we always deal with finite datasets, however ways to mitigate it have been developed.

One of the most attractive approaches to increase accuracy and/or minimize error variance (i.e. generate classification models which are more trustworthy to be safe from excessive overfitting) is the idea of combining classifiers. It has attained a vivid interest in the past decade, which resulted in a number of theoretical and practical achievements, especially in the domain of decision trees (for a survey see e.g. [9, 10]).

The success of ensembles can be explained in following words: if each particular voter (=component classifier) produces different approximation to real decision boundaries, then during the voting (which is kind of averaging process) the noise is supposed to be smoothed out, especially if the number of voters is large. This, however, holds true when the voters are independent. The condition of independence is not only hard to fulfill but even to measure in a real-life (finite) task. In practice we thus rather strive to select weakly correlated („diverse"), but still quite accurate classifiers. Several heuristic measures of diversity between component classifiers have been proposed and tested [11, 12], but any stronger guidelines on how to choose the classifiers for the ensemble still cannot be given.

Fig. 2 shows a case where an ensemble of five simple classifiers under majority voting solves the classic XOR problem. Note the components are not independent (e.g. classifiers 1 and 5 have same predictions on the whole domain).

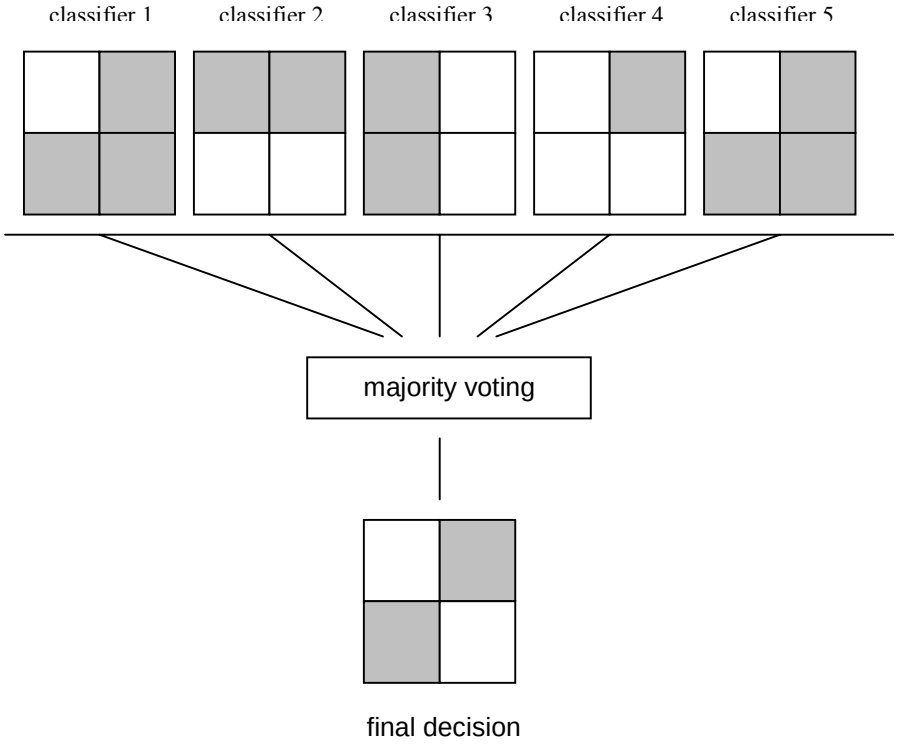

Figure 2: An ensemble of classifiers solves the XOR problem

Little has been done specifically in combining nearest neighbor classifiers. We are aware only of a few papers in this domain, namely Skalak's combining 1-NN classifiers with radically reduced prototype sets [8, 13], voting over Hart's condensed nearest neighbor classifiers [14], Multiple Feature Subsets (MFS) algorithm [15] and

decomposition of a multi-class problem into a net of dichotomizers [16, 17] (the last idea is of more general use and in fact has been known for years in neural network community [1]).

In this chapter we attempt to overcome the problem of selecting k in k-NN (or another k neighbors' based classifier) through using a number of k-NN (or such like) classifiers, each working on the whole reference set, but with its own value of k. Final decision is obtained via simple voting. The component k-NN classifiers are trained on random partitions of the whole learning set. Our goal is to increase accuracy rather than to decrease training costs (which in fact must be greater in our version compared to plain k-NN). Beneath we refer to the algorithm as to *voting* k-NN.

How does the learning in voting k-NN proceed in detail? A known technique which allows to estimate a classifier during the learning stage (i.e. in design time) is partitioning the whole learning set into two parts: a „real" training set and a validation set. The models for the classifier under design are applied for the former set but tested on the latter one. Of course, an obvious deficiency of such a technique is shrinking the training set. This, however, seems a price we must pay for the comfort of some estimation of the classifier yet in the learning stage. The question arises: how the given set should be divided into those two parts? Too few samples in the training set results in a very weak approximation of the underlying distribution. Too few samples in the validation set, on the other hand, implies an unreliable estimation of the generated classifier.

Our decision was to divide the learning set into halves. The optimal k for the training half was then sought with respect to the validation half. Such a random split followed with a k-NN learning session was performed L times to obtain L values: k_1, ..., k_L. The learning routine is also presented in Fig. 3 in a pseudo-Pascal code.

```
{ Tr – training set }
for i:=1 to L do  { L trials }
begin
  RS(i) := random_select(size(Tr) / 2);
  { RS(i) – random subset of Tr }
  CVS(i) := Tr \ RS(i);  { CVS(i) – current validation set }
  k(i):=find_best_k_for_kNN(RS(i), CVS(i));
  { on RS(i), with respect to CVS(i) }
end;
{ output: learnt parameters k(1), ..., k(L) }
```

Figure 3: Finding parameters for voting *k*-NN

The classification of a query sample q consists in producing L class labels for q according to k_i-NN, i=1..L, and finally assigning q to the class most frequently appearing among those L labels. By analogy we construct voting k-NCN and voting k-NSN rules.

4 Computational issues

What is the time complexity of finding k nearest neighbors? In practice, usually a *naive* implementation is used. During the search a sorted list of k NN's for a query sample is kept and updated when needed. Although typically (and for small enough values of k) the search time is about 1.1n..1.2n (i.e. only some 10-20% longer than 1-NN search time), in worst case it will be $O(d \cdot n \cdot k)$, d – dimension. To protect from the worst case, another

implementation can be used. First, distances to all n samples from the reference set are calculated and sorted, and then voting over classes of the first k neighbors incurs the classification decision. The classification time is therefore $O(d \cdot n + n \log n + k + c)$, d – dimensionality, c – number of classes. As $k \leq n$ and $c \leq n$ (typically $k << n$ and $c << n$), and usually log n is on the order of magnitude of d, the overall cost is close to $O(d \cdot n)$.

The classification in voting k-NN may follow the latter of the described implementations. The difference is in the final stage: instead of scanning k neighbors, we have to take into account $k_{\max} := \max_{i=1..L} k_i$ neighbors. The overall cost is $O(d \cdot n + n \log n + k_{\max} + L \cdot c)$. For reasonable values of L (in our tests L=10 seemed fairly good) the cost is practically comparable to the cost of original k-NN in worst-case protecting version. If the naïve neighbor search is used, then voting k-NN is slightly slower than plain k-NN, because k_{max} is often about twice greater than the globally selected k.

In the case of voting k-NCN, the slow-down related to the conventional rule is equal to the ratio of k_{max} and the globally selected k. The slow-down factor for voting k-NSN is close to an analogous ratio.

5 About the data

Our considerations concern images obtained by two sensors installed on an aircraft: a Daedalus 1268 Airborne Thematic Mapper scanner and a fully polarimetric PLC band NASA/JPL airborne radar system. The geographical location was the Feltwell area. The average registration error was on a pixel level. The ATM images were filtered by a linear smoothing and context-sensitive enhancement filter; then they were segmented by a multiband region-growing technique. Five following regions, i.e. classes, were selected: carrots, potatoes, stubble, sugar beet and wheat. Each pixel was described by 9 features, obtained from optical and radar channels. Below a brief feature description is given:
features 1 – 6 are responses of the Daedalus sensor for bands from 2 to 7 respectively, feature 7 is a response for the band C with HH polarization, feature 8 is a response of the radar sensor for L band and HV polarization and the feature 9 is a response of the radar sensor for the band P and VV polarization. More detailed description of the data can be found in [18].

Originally, the authors of [18] considered 15 features. However, our experiments have been constrained to the first 9 features out of 15. Our goal is to present rather the new approaches than search for the features, which would enable the smallest error rate. A data set, we deal with, contains 5 classes, 9 features and 5124 pixels.

6 Experimental comparison of the classifiers

We conducted experiments on real data taken from a remote sensing application described in the previous section.

Experiments were performed for training sets with sizes of 500, 750 and 1250 samples, each class represented with the same number of instances; the remaining samples formed respective test sets. For each training set size, ten partitions have been made; the presented results are averages over 10 runs. The city-block metric was used in all tests.

The k's for plain k-NN and k-NCN were found with leave-one-out cross-validation. As mentioned earlier, the k-NSN experiments benefited from learning sessions of k-NCN. The k-NSN based tests were run with the number of mutations for k-NSN sessions equal to 500 and 2500. All voting algorithms were tested in the described manner with 10 component classifiers. In all trials, both for plain and voting classifiers, the values of k were inspected in the interval 1..30. The results, in per cent, are presented in Table 1.

		k-NN	voting k-NN	k-NCN	voting k-NCN	k-NSN, 500 mut.	voting k-NSN, 500 mut.	k-NSN, 2500 mut.	voting k-NSN, 2500 mut.
size 500	error (%)	**23.1**	**23.4**	**23.5**	**22.7**	**22.8**	**22.0**	**22.9**	**22.2**
	st. dev. (%)	1.0	0.6	1.1	0.7	0.6	0.6	0.8	0.6
size 750	error (%)	**22.6**	**22.2**	**21.5**	**20.9**	**21.1**	**20.5**	**21.2**	**20.5**
	st. dev. (%)	0.5	0.6	0.9	0.6	0.7	0.5	0.6	0.5
size 1250	error (%)	**21.4**	**20.7**	**20.2**	**19.7**	**20.0**	**19.2**	**19.9**	**19.2**
	st. dev. (%)	0.8	0.9	0.9	0.5	0.8	0.5	0.6	0.4

Table 1: Comparison of described algorithms on remote sensing data

It should be noticed that the described voting technique decreased the test errors of all respective base classifiers in all cases, and also decreased the error variance in most cases. The average errors for the most successful classifier (i.e. voting k-NSN with 500 mutations) were lower than the errors offered by the plain k-NN rule by more than 2%.

In Table 2 we compare the numbers of NCN neighbors used for each partition of the datasets and the maximal numbers of NCN neighbors in the voting scheme. For example, the ratio of about 1.6 means exactly that on the given dataset the voting algorithm is on the average slower than plain k-NCN by the factor of about 1.6.

size	classifier case	partition number										mean	ratio k_{max} / global k
		01	02	03	04	05	06	07	08	09	10		
500	k in plain k-NCN	11	10	21	14	7	10	15	21	11	17	13.7	1.6
	k_{max} in voting k-NCN	19	26	24	27	25	13	20	19	20	22	21.5	
750	k in plain k-NCN	14	11	11	14	15	10	21	7	20	10	13.3	1.4
	k_{max} in voting k-NCN	17	28	16	29	12	14	20	13	20	17	18.6	
1250	k in plain k-NCN	15	13	11	14	27	7	11	12	14	13	13.7	1.5
	k_{max} in voting k-NCN	22	15	24	18	16	19	21	29	23	20	20.7	

Table 2: The values of k selected for each partition

7 Remarks about the k-NN, k-NCN and k-NSN classifiers

The goal of our work was to create a homogeneous ensemble of k-NN-like classifiers which would differ only in the number k of neighbors used for prediction by each

component. The ensemble was expected to be less prone for overfitting the training data and an additional motivation was a modest increase of the classification time.

We obtained quite promising results. There is some resemblance between our method and the old idea of weighted *k*-NN [19, 20]. Nearer neighbors generally affect more component decisions than farther ones. This is similar (but not equivalent) to setting weights. One difference to the referenced concept must be however stressed: our „weights" are rank-based, not distance-based. Although the pointed effect may shed some light on why the scheme can work, definitely much more insight is required.

In our previous paper [21] we conjectured the introduced voting idea may be applied also to classifiers more complex than *k*-NN. Indeed, the results of voting *k*-NCN and *k*-NSN presented here confirm attractiveness of the approach. The classification speed decrease, really small in the voting *k*-NN case, is however greater now, reaching in our experiments about 50% penalty with *k*-NCN. This, however, still contrasts with most ensemble methods, where the slow-down compared to a single component classifier is usually proportional to the number of components.

We have not tried to combine the voting idea with any pairwise scheme for multidecision tasks [16, 22]. This is going to be a subject of our future experiments.

A separate contribution of this paper are further tests of the *k*-NCN and *k*-NSN rules. The surrounding neighborhood concept and in particular its Nearest Centroid realization, poses several interesting questions:

- Does the centroid criterion reflect the neighborhood homogeneity really well? (Note a set of points (=neighbors) on a line could have a gravity center exactly at the test sample *q*, but intuitively we would not call this set as lying symmetrically around *q*.)
- Even with the centroid criterion in mind, are there possible other efficient methods of optimizing the neighborhood?
- What about voting over several (different) surrounding neighborhoods?
- For a given set, is it possible to predict if a given method succeeds or fails?

Of course, the last question is of much more general importance.

8 Reference set size reduction problem

The classifiers for remote sensing problems are usually constructed with the use of the large date sets. For this reason the classification speed may be not satisfactory. The most promising way to make the classification faster consists in reference set reduction. It is an interesting problem how to reduce the reference set or to replace it by a smaller one without a remarkable decrease of the classification quality. Numerous effective reference set reduction algorithms have been devised only for the 1-NN rule that usually yields worse performance as compared to the standard *k*-NN rule. For this reason the classical *k*-NN rule may first be approximated by the 1-NN rule. To do this, it is sufficient to reclassify the original reference set, i.e. training set, and then to reduce it and use with the 1-NN rule. However, it is also worth to check how behaves the *k*-NN rule operating with the reduced sets obtained for the 1-NN rule.

Three most popular reference set reduction [23, 24, 25] algorithms based on consistency idea will be compared with the approach that consists in reference set partitioning. The consistency means that 1-NN rule operating with the reduced set classifies correctly all points from the original reference set. These algorithms determine the size of the reduced set. The procedure based on reference set partitioning described in

[26] can also define the size of the condensed reference set. The term *condensed* is used to stress that the obtained set is not a subset of the primary reference set. A slight modification of this procedure allows to condense the reference set to the desired size. In the next two sections the detailed description of the analyzed methods of the reference set size reduction will be given.

9 Reference set reduction algorithms

Hart's algorithm. The first point from the reference set is qualified to the (initially empty) reduced reference set. Next, the remaining points of the primary reference set are classified by the 1-NN rule with the current reduced reference set. Each misclassified point is added to the reduced reference set. Such classification of all points from the primary reference set is repeated as long as m subsequent classifications do not increase the size of the reduced reference set. The first points selected to the reduced reference set can lie far away from the class boundary. This disadvantage of Hart's algorithm has been removed by Gowda-Krishna modification. The Hart algorithm was originally proposed in the paper [23].

***Gowda-Krishna's* algorithm.** A mutual distance measure $mdm(\mathbf{x})$ is associated with each point $\mathbf{x}$ of the primary reference set. The $mdm(\mathbf{x})$ is calculated in the following way. For the point $\mathbf{x}$ the nearest point $\mathbf{y}$ from the opposite class is found. A number of points from the same class as $\mathbf{x}$ that lie closer to $\mathbf{y}$ than $\mathbf{x}$ to $\mathbf{y}$ is the value of $mdm(\mathbf{x})$. Next, all the points of the primary reference set are arranged according to growing values of $mdm(\mathbf{x})$. Finally, the Hart algorithm is applied to the reference set ordered in this way. This modification of Hart's algorithm was proposed in the work [24].

***Tomek's algorithm*.** Each point $\mathbf{x}$ for which exists a point $\mathbf{y}$, from another class than $\mathbf{x}$, such that the internal part of the ball spanned by the points $\mathbf{x}$ and $\mathbf{y}$ does not contain any points from the reference set, is qualified to the reduced reference set. Originally [25], this algorithm was defined in a different way. Furthermore, the two class problem and the Euclidean distance function was assumed.

The authors of the three above described algorithms tried to construct the so called *consistent reduced reference set*, *i.e.* the set which, when used as the reference set with the 1-NN rule leads to correct classification of all points from the primary reference set. However, in the case of Tomek's algorithm the consistency is not guaranteed.

Gowda-Krishna algorithm produces the smallest size of the reduced set, the Hart's algorithm is the fastest and the Tomek's procedure generates the separating hypersurfaces close to the ones based on the whole reference set.

10 Reference set condensation algorithm

To describe the algorithm it will be convenient to introduce the notion of a *diameter* of the set, which is understood as the Euclidean distance between its two farthest points.

***Condensation by multiple reference set partitioning*.** At the start point the condensed set contains only one point $\mathbf{G}(1)$, equal to the reference set gravity center and labeled as the majority of its points. So, the label of this point corresponds to the class most heavily represented in the training set. Then two farthest points $\mathbf{P}_1$ and $\mathbf{P}_2$ of the reference set are found. The first partition is performed by a hyperplane passing in the middle between $\mathbf{P}_1$ and $\mathbf{P}_2$ and orthogonal to straight line that joins these points. The points, which lie closer to the point $\mathbf{P}_1$ than to the point $\mathbf{P}_2$ or in the same distance form

the set **C**(1) and the remaining points create the set **C**(2). The previous point **G**(1) is replaced by the gravity center of **C**(1). The point **G**(1) and the gravity center **G**(2) of **C**(2) assume the labels as the majority of points in the sets **C**(1) and **C**(2) respectively. Now, the condensed set contains two points, **G**(1) and **G**(2).

Let us assume that the original reference set has been dropped into m-1 subsets **C**(i). The current condensed set contains then m-1 points **G**(i), gravity centers of **C**(i). From among all **C**(i) containing at least two points from different classes the set **C**(j) with the largest diameter is selected. This set with the help of its two farthest points $\mathbf{P}_1$ and $\mathbf{P}_2$ is divided into two parts $\mathbf{D}_1$ and $\mathbf{D}_2$. The old C(j) is replaced by $\mathbf{D}_1$ and the new set **C**(m)=$\mathbf{D}_2$ is created. Thus, the number of subsets C(i) has been increased from m-1 to m. Taking new **G**(j) as the gravity center of $\mathbf{D}_1$ and computing **G**(m) as the gravity center $\mathbf{D}_2$, the size of the current condensed set will be increased to m. In this way, by virtue of the described recursion scheme, the number of subsets can get larger and larger until all **C**(i) with at least two points from different classes will be exhausted. So, the size of the condensed reference set is determined. A more formal description of this algorithm can be found in [26].

11 Computational results

Five hundred pixels (one hundred from each class) were randomly selected to be used as the training set, and the remaining 4624 pixels were treated as the testing set. Such an experiment was repeated 10 times. In each experiment all the algorithms presented in the previous two sections were applied.

The optimum value of k was found first for the whole training set by use of the *leave one out* method. If more then one value of k offered the smallest error rate then largest one was selected since such a choice promises a lower standard deviation of the misclassification rate. Then this k was used to reclassify all ten training sets. The raw and the reclassified training sets were separately applied for the reduced and condensed reference set constructions. The values of k for the reduced (condensed) sets were calculated also by the *leave one out* method only on the basis of these sets. Error rates were calculated by use of corresponding testing sets. Results of the computations were gathered in Table 3. The most interesting results have been marked by the bold font.

One can notice that Tomek's algorithm gives very weak reduction. The error rates are nearly the same as the ones offered by the complete reference set, but the reduction degree is very small. The use of 1-NN rule with the reclassified set instead of k-NN with raw data is fruitful. The former results in 205 points in the reduced set and error rate equal 26.5%, while the latter produces 294 points and the misclassification rate equal 32.9%. No significant difference is observed between the Hart and the Gowda-Krishna procedures.

Remarkably better results, taking into account the reduction degree as well as the error rate, promises the approach based on reclassified reference set dividing. It offers twice smaller reference set size reduction saving nearly the same performance as Tomek's algorithm. The solution with k-NN rule, k>1, costs slightly more time in comparison to the case when k=1. A suitable modification of the well known Quick Sort algorithm allows very fast search of the nearest neighbors.

The result of Hart's and Gowda-Krishna's approaches are also worth of attention because of strong reduction.

Decision rule →	Reference set						
	Original					Reclassified	
	1-NN		*k*-NN			1-NN	
Reduction type ↓	m_{red}	*error rate %*	*k*	m_{red}	*error rate %*	m_{red}	*error rate %*
Column 0	1	2	3	4	5	6	7
No reduction, mean value	500	30.0	**9**	**500**	**23.1**	500	26.1
Standard deviation	*0*	*1.0*	*2*	*0*	*0.8*	*0*	*0.6*
Tomek, mean value	473	29.9	8	473	23.5	419	26.1
Standard deviation	*4.9*	*0.9*	*3*	*4.9*	*1.0*	*12*	*0.6*
Hart, mean value	255	34.7	15	255	27.4	**162**	**28.3**
Standard deviation	*10*	*1.0*	*4*	*10*	*2.4*	*11*	*1.3*
Gowda-Krishna, mean value	245	35.0	26	245	29.3	**149**	**28.9**
Standard deviation	*12*	*0.6*	*12*	*12*	*2.9*	*11*	*1.3*
Partitioning, mean value	294	32.9	**11**	**294**	**25.1**	**205**	**26.5**
Standard deviation	*13*	*1.0*	*5*	*13*	*1.5*	*14*	*1.2*

Table 3: Results of reference set size reduction

12 Conclusions

As it was explained in Section 1, the classifiers based on *k*-NN rule promise the performance close to the one offered by the Bayes classifier if the training sets are sufficiently large. We have shown that some modifications of the *k*-NN rules can outperform the original standard version, it depends on the data we deal with. The larger are the training sets, the smaller difference between the standard and the modified version of the NN type classifier can be expected. We feel that also the performance of the *k*-NN rules or 1-NN rule operating with the reduced or the condensed sets converge to the performance of the Bayes classifier. Thus, for the very large data there will be no reason to use the reference set of the original size.

In the case of the rules based on the reduced reference sets we have used the standard *k*-NN rule with the raw data or 1-NN rule with the reference sets reclassified by the *k*-NN rule. It would be interesting to examine the results, which could be obtained by association the reduced reference sets with the proposed *k*-NSN classifier. The classifier for remote sensing problems can be based on several time larger data that the ones used in the present paper. The reference set reduction can not only accelerate the classification phase but also the training stage.

It is worth to notice that reference set condensation algorithm starts with one point in the condensed set and stops when each subset of the original reference set contain points from the one class only. So, there is a possibility to control the classification quality after each sequential increasing of condensed set by one point. In this way we can find the most appropriate compromise between the speed and the quality of the classification phase.

13 Acknowledgement

The support of NATO Grant PST.CLG.977258 on this work is gratefully acknowledged.

References

1. Duda R. H., Hart P. E., *Pattern Classification and Scene Analysis,* John Wiley, New York, 1973.
2. Fix E., Hodges J. L., 1952, "Discriminatory Analysis: Nonparametric Discrimination Small Sample Performance," *project 21-49-004, Report Number* 11, USAF School of Aviation Medicine, Randolph Field, Texas, pp. 280–322.
3. Devijver P. A., Kittler J., *Pattern Recognition: A Statistical Approach,* Prentice Hall, London, 1982.
4. Chaudhuri B. B., 1996, "A new definition of neighbourhood of a point in multi-dimensional space," *Pattern Recognition Letters*, Vol. 17, pp. 11–17.
5. Sánchez J. S., Pla F., Ferri F. J., 1997, "On the use of neighbourhood-based non-parametric classifiers," *Pattern Recognition Letters*, Vol. 18, pp. 1179–1186.
6. Sánchez J. S., Pla F., Ferri F. J., 1998, "Improving the *k*-NCN classification rule through heuristic modifications," *Pattern Recognition Letters*, Vol. 19, pp. 1165–1170.
7. Grabowski Sz., 2002, "Towards decision rule based on closer symmetric neighborhood," *Biocybernetics and Biomedical Engineering*, accepted.
8. Skalak D. B., 1994, "Prototype and Feature Selection by Sampling and Random Mutation Hill Climbing Algorithms," *Machine Learning: Proceedings of the Eleventh International Conference*, Morgan Kaufmann.
9. Dietterich T. G., 2000, "Ensemble methods in machine learning," *Multiple Classifier Systems. First International Workshop (MCS2000)*, Cagliari, Italy: Springer-Verlag, pp. 1–15.
10. Bauer E., Kohavi R., 1999, "An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants," *Machine Learning*, 36, pp. 105–142.
11. Kuncheva L. I., Whitaker C. J., 2002, "Measures of diversity in classifier ensembles," *Machine Learning*, accepted (available at http://www.bangor.ac.uk/_mas00a/papers/lkml.ps.gz).
12. Shipp C. A., Kuncheva L. I., 2002, "Relationships between combination methods and measures of diversity in combining classifiers," *Information Fusion*, Vol. 3, pp. 135–148.
13. Skalak D. B., 1996, "Prototype Selection for Composite Nearest Neighbor Classifiers," PhD thesis, Dept. of Computer Science, University of Massachusetts.
14. Alpaydin E., 1997, "Voting over multiple condensed nearest neighbors," *Artificial Intelligence Review*, Vol. 11 (1-5), pp. 115–132.
15. Bay S. D., 1998, "Combining Nearest Neighbor Classifiers Through Multiple Feature Subsets," *Proceedings of the Fifteenth International Conference on Machine Learning*, Madison, WI.
16. Jozwik A., Vernazza G., 1988, "Recognition of leucocytes by a parallel *k*-NN classifiers," *Lecture Notes of ICB Seminar*, Warsaw, Poland, pp. 138–153.
17. Ricci F., Aha D. W., 1998, "Error-correcting output codes for local learners," *Proceedings of the Tenth European Conference on Machine Learning*, Chemnitz, Germany.

18. Serpico S. B., Roli F., 1995, "Classification of multisensor remote sensing images by structured neural networks," *IEEE Trans. Geoscience Remote Sensing*, Vol. 33, No. 3, pp. 562–578.
19. Dudani S. A., 1976, "The Distance-Weighted k-Nearest Neighbor Rule," *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 6, No. 4, pp. 325–327.
20. Wilson D. R., Martinez T. R., 2000, "An Integrated Instance-Based Learning Algorithm," *Computational Intelligence*, Vol. 16, No. 4, pp. 1–28.
21. Grabowski Sz., 2002, "Voting over Multiple k-NN Classifiers," *Proceedings of the IEEE Conference TCSET'2002*, Lviv-Slavsk, Ukraine, pp. 223–225.
22. Moreira M. Mayoraz E., 1998, "Improved pairwise coupling classification with correcting classifiers," *Proceedings of the Tenth European Conference on Machine Learning*, Chemnitz, Germany.
23. Hart P.E., 1968, "The condensed nearest neighbor rule," *IEEE Trans. Information Theory*, Vol. 14, No. 3 (Corresp.), pp. 515–516.
24. Gowda K. C., Krishna G., 1979, "The condensed nearest neighbor rule using the concept of mutual nearest neighborhood," *IEEE Trans. Information Theory*, Vol. 25, No. 4, pp. 488–490.
25. Tomek I., 1977, "Two modifications of CNN," *IEEE Trans. Systems, Man, and Cybernetics*, Vol. 7, No. 2, pp. 92–94.
26. Chen C. H., Jóźwik A., 1996, "A sample set condensation algorithm for the class sensitive artificial neural network," Vol. 17, pp. 819–826.