Pattern Recognition and String Matching
D. Chen and X. Cheng (Eds.) pp. – - –

Enhanced Neighbourhood Specifications for Pattern Classification

J. S. Sánchez
Department of Computer Languages and Information Systems
University Jaume I, Av. Vicent Sos Baynat S/N, 12006 Castellón, Spain
E-mail: sanchez@uji.es

A. I. Marqués
Department of Business Administration and Marketing
University Jaume I, Av. Vicent Sos Baynat S/N, 12006 Castellón, Spain
E-mail: imarques@uji.es

Contents

1 Introduction

Automatic classification is an active research area in artificial intelligence, pattern recognition, and machine learning. Its goal is to make a computer automatically label attribute vectors as examples of two or more different categories or classes. One of the most widely studied approaches corresponds to the well known Nearest Neighbour (NN) decision rule, proposed by Fix and Hodges [17], formally analysed by Cover and Hart [8], and then investigated both theoretically and experimentally by many researchers. On the other hand, a lot of successful applications have been reported in a multitude of real-world domains [9].

In its classical manifestation, given a set of n previously labelled prototypes (attribute vectors) or training set (TS), the NN classifier assigns any new pattern to the class indicated by the label of the most *similar* prototype in the TS. The degree of similarity between two samples is established by means of appropriate distance metrics defined on the feature space. Thus, the smaller the distance between the two attribute vectors, the higher their mutual similarity.

A slight improved version of the NN rule consists of considering k neighbours of the pattern to classify. Then, a new sample is assigned to the class most frequently represented among the k closest prototypes in the TS, according to a certain similarity measure. Apart from their conceptual and implementational simplicity, the NN decision rules are optimal under the theoretical assumptions of infinite number of training prototypes (that is, when $n \to \infty$). Cover and Hart [8] demonstrated that the asymptotic classification error rate of the k-NN rule tends to the optimal Bayes error as

$k \to \infty$ and $k/n \to 0$. Moreover, if $k = 1$, the NN error is bounded by approximately twice the Bayes error [13].

Nevertheless, in many practical settings, such a theoretical behaviour can hardly be achieved because of certain inherent weaknesses that significantly reduce the applicability of k-NN classifiers in real-world tasks. For example, memory requirements can be prohibitive in domains with a lot of prototypes (e.g., in data mining applications), and also classification costs (in terms of response time) are too high if the decision rule has to compare a given unclassified pattern with millions of training examples.

Several works have focused on alleviating the high computational and storage loads of these classifiers by means of fast NN search algorithms [3, 14, 48]. Other techniques, namely *condensing*, try to minimize these practical problems by selecting a sufficiently small subset of prototypes that leads to approximately the same performance than the NN rule using the whole TS [10, 18, 21, 44].

Another drawback for NN classifiers refers to the fact that the TS may contain noisy, mislabelled and atypical prototypes, which generally lead to a significant decrease in performance. Wilson [49] proposed the first algorithm to tackle this problem: it utilizes the k-NN rule to estimate the class label of all prototypes in the TS and discards those whose label does not agree with the class associated with the largest number of the k neighbours. Subsequently, important research has been carried out to deal with this critical issue [5, 13, 22, 31, 38] both in pattern recognition and machine learning domains. In general, *editing* algorithms try to eliminate erroneously labelled prototypes from the original TS and "clean" possible overlapping among regions from different classes, usually yielding significant improvements in classification performance.

Joint use of editing and condensing tools, generally referred as to *prototype selection* techniques [13], leads to a composite approach in which both computational efficiency and classification accuracy are improved [11]. In fact, editing and condensing are two closely related and complementary procedures: condensing makes sense only when the classes are clustered and well-separated, which constitutes the aim of the editing algorithms.

On the other hand, NN classification accuracy significantly drops in environments where many attributes describing the prototypes are irrelevant; such attributes or features inappropriately affect the values returned by most similarity metrics. Langley and Iba [33] show that the number of prototypes needed to reach given accuracy grows exponentially with the number of irrelevant attributes. A considerable amount of research has been addressed

to reduce sensitivity of NN rules to irrelevant features by selecting a subset of relevant attributes [4, 25, 29, 32, 43] and also by assigning different weights to each feature [1, 7, 27, 34].

Recently, significant research in NN classification has focused on using distinct specifications of the neighbourhood concept inherent in all similarity-based classifiers. The ultimate goal of such alternative neighbourhoods is to overcome some of the aforementioned problems related with NN classification. In this context, several empirical studies [38, 39, 45, 51] have shown that in many cases, the degree of similarity among the prototypes in a TS cannot be appropriately estimated by taking into account only some distance measure but also considering other geometrical relationships among them.

This chapter concentrates on an alternative neighbourhood concept, namely *surrounding neighbourhood*, proposed in the last years and applied to very different domains. We focus primarily on contributions to pattern classification and prototype selection. Accordingly, Section 2 introduces the rationale of surrounding neighbourhood and describes some particular implementations. The following two sections report practical applications of such a neighbourhood realization in the framework of pattern recognition. Finally, Section 5 includes the main points of this chapter and some comments on topics for further research.

2 Surrounding Neighbourhood

In this section, we explore the general concept of neighbourhood in the context of pattern classification and compare two alternative definitions: the plain or nearest neighbourhood (that is, neighbourhood defined from the minimum distance) and the novel surrounding neighbourhood. Furthermore, some particular realizations of the surrounding neighbourhood are here reported.

The concept of neighbourhood has widely been used in a large number of robotics, computer vision, pattern recognition, computational geometry and image processing problems [12, 20, 24, 50]. The neighbours of an input point p in a d-dimensional feature space can be loosely viewed as the set of points located in the proximity of p. In fact, the neighbours of a certain point p are usually assumed as those points that are the closest to p in terms of Euclidean or other norm-based distance metric.

However, neighbourhood can intuitively be understood as a property

subject to two complementary constraints. Firstly, the neighbours of a point p should be as *close* to it as possible. And secondly, the neighbours should also be located as *symmetrically* around p as possible. As the nearest neighbourhood takes only the first condition into account, the neighbours selected may not be well distributed around the test point in the feature space. In general, when the neighbours of a point are homogeneously distributed around it, it seems that more local information is provided to appropriately classify the given point.

A number of different approaches have already been proposed in order to undertake this problem [6, 36, 46, 51]. All these methods have in common the fact of considering both proximity and symmetry so as to define the general concept of neighbourhood. Thus, they try to search for neighbours close enough (in the basic distance sense), but also homogeneously or symmetrically distributed around a given point. For this reason, some researchers have recently referred to this heuristic concept as *surrounding neighbourhood* [39, 51]. In particular, this chapter introduces three different approaches to such a neighbourhood representation, that is, the graph neighbourhood [38], the nearest centroid neighbourhood [6], and the Zhang's neighbourhood [51].

2.1 Graph Neighbourhood

Let $X = \{x_1, \ldots, x_n\}$ be a finite set of n points in R^d, where d denotes the dimensionality of the feature space. From a general point of view, a proximity graph, say $G = (V, E)$, is defined as an undirected graph with the set of vertices $V = X$, and the set of edges, E, such that $(x_i, x_j) \in E$ if and only if x_i and x_j satisfy some mutual neighbourhood property. In such a case, it is said that x_i and x_j are *graph neighbours*. Thus the set of graph neighbours of a given point constitutes its *graph neighbourhood*. The graph neighbourhood of a subset, $S \subseteq V$, consists of the union of all the graph neighbours of every node in S.

The Gabriel Graph (GG) and the Relative Neighbourhood Graph (RNG), along with the Delaunay Triangulation (DT) and the Minimum Spanning Tree (MST) are probably the most prominent types of proximity or closest-point graphs [37]. As an important property, the DT is a supergraph of the GG, this constitutes a supergraph of the RNG, and this is a supergraph of the MST. Thus, they form the following hierarchical relationship:

$$MST \subseteq RNG \subseteq GG \subseteq DT$$

It is also important to remark that, with only minor modifications to the algorithms for computing the DT, any one of the other graphs can be easily and efficiently constructed [47]. Jaromczyk and Toussaint [26] reported a complete survey of the most relevant properties of these and other proximity graphs.

The DT of a set of points X is defined as the dual graph of the *Voronoi diagram* associated with the set X, which constitutes a decomposition of R^d into a number of convex regions or *Voronoi cells*, each of which defines the "region of influence" of a given point in its interior [37]. A Voronoi cell is a d-dimensional polytope that can be defined as the intersection of a finite number of closed half-spaces and is therefore, supported by a finite number of hyperplanes. Consider all triangles formed by the points such that the circumcircle of each triangle is empty of other points. The set of edges of these triangles gives the DT of the points. Therefore, two points in a DT are connected with an edge if and only if the boundaries of their Voronoi cells are adjacent.

In this chapter, we concentrate our examination only on the GG and the RNG because it has recently been done considerable work on the application of these particular proximity graphs to several classification problems. In fact, it has been demonstrated that they generally obtain a higher classification accuracy than other conventional NN-based approaches. On the other hand, there exist more efficient algorithms for the computation of the GG and the RNG in high dimensional spaces than for the construction of the DT [26, 37].

The complexity of a brute-force algorithm [45] for computing the GG (or the RNG) over an input data set with n points in d-space is $\mathrm{O}(dn^3)$, which is absolutely prohibitive for all practical purposes. There also exists a heuristic method [45] with an expected computational cost close to $\mathrm{O}(dn^2)$, although it has to be admitted that this is still of little value for several real-time applications.

2.1.1 Gabriel Graph

Let $d(\cdot,\cdot)$ be the Euclidean distance between two points in R^d, then the set of edges in a GG is defined as follows:

$$(x_i, x_j) \in E \Leftrightarrow d^2(x_i, x_j) \leq d^2(x_i, x_k) + d^2(x_j, x_k) \quad \forall x_k \in X, k \neq i, j$$

In this case, x_i and x_j are said to be *Gabriel neighbours* (GN). In other words, two points are GNs if and only if there is no other point from X

laying in a hypersphere Γ_{x_i,x_j} (namely, diameter hypersphere) centered at their middle point and whose diameter is equal to the distance between them.

Let $B(p,r)$ denote an open hypersphere centered at p with radius r, i.e., $B(p,r) = \{q : d(p,q) < r\}$, then the diameter hypersphere Γ_{x_i,x_j} can be defined as:

$$\Gamma_{x_i,x_j} = B(\frac{x_i + x_j}{2}, \frac{d(x_i,x_j)}{2})$$

Now, the set of edges in a GG over a set of points X can be expressed using an equivalent definition as follows:

$$(x_i,x_j) \in E \Leftrightarrow \Gamma_{x_i,x_j} \cap X = \emptyset$$

From the set of GNs of a given point p, it is possible to define a surrounding region, here called the *Gabriel surrounding neighbourhood*, Γ_p, as the union of the diameter hyperspheres between p and all its GNs, x_i:

$$\Gamma_p = \cup\Gamma_{p,x_i} \quad \forall x_i \in X \ / \ \Gamma_{p,x_i} \cap X = \emptyset$$

2.1.2 Relative Neighbourhood Graph

In a similar fashion, the set of edges in a RNG can be obtained in the following way:

$$(x_i,x_j) \in E \Leftrightarrow d(x_i,x_j) \leq max\,[d(x_i,x_k), d(x_j,x_k)] \quad \forall x_k \in X, k \neq i,j$$

Now its corresponding geometric interpretation is based on the concept of lune Λ_{x_i,x_j}, which is defined as the disjoint intersection between two hyperspheres centered at x_i and x_j and whose radii are equal to the distance between them.

$$\Lambda_{x_i,x_j} = B(x_i, d(x_i,x_j)) \cap B(x_j, d(x_i,x_j))$$

Thus two points are *relative neighbours* (RN) if and only if their lune does not contain other points of the set X in its interior.

$$(x_i,x_j) \in E \Leftrightarrow \Lambda_{x_i,x_j} \cap X = \emptyset$$

In this case, the *relative surrounding neighbourhood* for a given point p, say Λ_p, can be defined as the union of the lunes between p and all its RNs, x_i:

$$\Lambda_p = \cup\Lambda_{p,x_i} \quad \forall x_i \in X \ / \ \Lambda_{p,x_i} \cap X = \emptyset$$

2.2 Nearest Centroid Neighbourhood

A second approach to surrounding neighbourhood comes from the concept of *nearest centroid neighbourhood* [6], which was initially applied to detect the border points and the interior points of a dot pattern, as well as to select a representative subset from a set of data.

In plain words, the nearest centroid neighbourhood concept is based on the direct application of those constraints mentioned at the beginning of this section. That is, by the *proximity criterion*, the k neighbours of a sample p must be as near as possible. At the same time, by the *symmetry criterion*, their centroid must be also as close to p as possible.

Let p be a given point whose k nearest centroid neighbours (NCN) should be found in a set of points $X = \{x_1, \ldots, x_n\}$, then the NCNs can be obtained as follows [6]:

1. The first NCN of p is its NN, say q_1.
2. The i-th neighbour, q_i, $i \geq 2$, is such that the centroid of this and all previously selected neighbours, $q_1, \ldots, q_{i-1}$ is the closest to p.

This iterative procedure clearly does not minimize the distance to the centroid because it gives precedence to the individual distances instead. However, because of the centroid criterion, the spatial distribution of neighbours is taken into account. On the other hand, proximity of the k NCNs to the input point is guaranteed due to the incremental nature of the way in which they are obtained from the first NN.

Let q_1 be the NN of a given point p, and let $C(S, y)$ denote the centroid of the set S and the point y. Then, the *centroid surrounding neighbourhood* for a given k, say $\Upsilon_k(p)$, can be defined as the set of prototypes that satisfy the following conditions:

1. $\Upsilon_1(p) = \{q_1\}$.
2. $\Upsilon_k(p) = \Upsilon_{k-1}(p) \cup \{x_i\}$, such that $d(p, C(\Upsilon_{k-1}(p), x_i)) \leq d(p, C(\Upsilon_{k-1}(p), x_j))$, $\forall x_i, x_j \in X - \{\Upsilon_{k-1}(p)\}$, $i \neq j$.

This kind of surrounding neighbourhood can be efficiently computed in $O(kn)$ time in any dimension. More specifically, the computation of one NCN requires at most n centroid and distance computations as well as n comparisons to find the minimum of the distances [6].

2.2.1 Nearest Centroid Neighbourhood with Local k

The greedy nature of obtaining successive NCNs broadens the search space, still with the centroid criterion in mind, even for the price of increasing computational requirements. This however seems difficult without an adverse effect on the proximity criterion.

Grabowski [19] has recently proposed a slight modification of the original concept of nearest centroid neighbourhood as a solution to such a relative drawback. For a given pattern p, all its k NCNs are computed and then, the first $k(p) \leq k$ neighbours, whose distance from the centroid $c_{k(p)}$ to p is minimal over all distances from centroids c_j, $(j = 1, \ldots, k)$ to p, are selected. In other words, this modification aims at minimizing the centroid criterion in a local manner.

2.3 Zhang's Neighbourhood

The surrounding neighbourhood implementations described in previous sections come from the pattern recognition domain. Nevertheless, researchers from the machine learning area have also utilised a similar concept in various applications. For example, Zhang et al. [51] proposed an algorithm to compute the so-called k surrounding neighbours of a given point. Here, this particular realization is called *Zhang's neighbourhood* so that it can be distinguished from the other surrounding representations.

The idea of the Zhang's approach is to select k prototypes that are not only close to a given sample, but also well distributed around it in the feature space. In other words, all selected neighbours should be close to the given prototype, but they should not be too close to each other. As can be seen, this rationale matches the concepts of graph neighbourhood and nearest centroid neighbourhood, in the sense of trying to select a set of prototypes widely distributed around the given point.

Let p be a new point whose k Zhang's neighbours (ZN) should be selected from a set of points $X = \{x_1, \ldots, x_n\}$, then these ZNs can be obtained as follows [51]:

1. The first ZN of p is its NN, say z_1.
2. The i-th neighbour, z_i, $i \geq 2$, is such that satisfies the following two conditions:
   - $d(z_i, z_{i-1}) > d(p, z_i)$
   - $d(x_j, p) \geq d(p, z_i) \quad \forall x_j \in X - \{z_k\},\ k = 1, \ldots, i$

The first statement implies that at each iteration, the two selected prototypes are not too close to each other, and the second condition means that they are "sufficiently" close to the new point p. As can be seen from the above procedure, like in the case of the NCN algorithm, proximity of the k neighbours to the input point is guaranteed because the first ZN is also selected from the first NN.

3 Classification Methods

Like the traditional nearest neighbourhood concept, the surrounding approaches can be applied to a number of pattern classification problems. In fact, the ultimate goal of the surrounding neighbourhood is to overcome some of the practical drawbacks outlined for the NN-based techniques and more specifically, to outperform the classification accuracy of such decision rules. Thus, this section summarises some classification schemes derived from the different realizations of surrounding neigbourhood introduced in Section 2.

The general idea behind these classifiers is to make a decision about the class membership of a given unclassified pattern after knowing the geometrical distribution of the training prototypes around it on the feature space. It must be pointed out that, even though there is no theoretical justification to use neighbours around (and also close) to a sample instead of NNs, this can effectively help in many practical situations (e.g., finite sample size case), in which prototypes do not fully represent the underlying statistics and/or the particular distance employed (irrelevant in the asymptotic case) exhibits some undesirable properties.

3.1 Classifiers Based on Proximity Graphs

Let $X = \{x_1, \ldots, x_n\}$ be a set of n training prototypes in a d-dimensional feature space, belonging to L different classes, and let q a new pattern to classify, then a classifier based on some kind of proximity graphs can be expressed as follows [39]:

1. Search for all the graph neighbours of q, say $X^q = \{x_1^q, \ldots, x_m^q\}$, where $m \leq n$.
2. Assign q to the class with a majority of votes from its m graph neighbours in the set X^q (resolve ties randomly).

Note that, considering the GG and the RNG, we have two different but analogous approaches, namely GN and RN decision rules, respectively. On the other hand, it is worth mentioning that the corresponding classifiers with a reject option [23] and other extensions [13] could also be defined exactly in the same way as in the k-NN rule.

It has empirically been demonstrated that the GN and RN classifiers generally outperform the k-NN decision rule, specially in domains where the number of training prototypes is not very large [39]. The reasons of such a fact would be two-fold: the classifiers based on this kind of surrounding neighbourhood seem to be more reliable because the estimate criterion employed takes into account the distribution of neighbours around the given unclassified sample and secondly, the computational cost to construct any of these graphs is still too high to consider the corresponding GN or RN rules in problems with a very large TS.

3.2 k-NCN Classification Rules

Several classification schemes derived from the nearest centroid neighbourhood have been proposed. The aim of using this kind of surrounding neighbourhood matches the general goals of the classifiers based on proximity graphs. In fact, the k-NCN decision rule differs from the GN and RN classifiers only in the neighbourhood criterion adopted.

This general strategy has been performed in a wide variety of implementations. Nevertheless, all of them primarily focus on the geometrical distribution of k neighbours around the test sample, instead of their distance to it.

The first and most trivial application of the nearest centroid neighbourhood to classification refers to the so-called k-NCN rule [39]. Let q be a certain pattern to classify, then the plain k-NCN approach can be defined in the following way:

1. Find the k NCNs of q.
2. Assign q to the class with a majority of votes among its k NCNs (resolve ties randomly).

Several examples of its practical employment in different real-world problems [15, 35] have demonstrated a certain increase in performance compared with the classification accuracy achieved by the conventional k-NN decision rule.

Just like in the case of the k-NN rule, also for the k-NCN classifier the

number of neighbours k must be estimated with respect to each particular TS, preferably by means of the leaving-one-out method.

3.2.1 Averaged k-NCN Rule

A first extension to the k-NCN classification rule is meant to resolve the k-NCN ties [40]. The "tie-breaking" modification can be defined as follows: a given unclassified pattern is assigned to the class that has received at least a half of the k votes (say, $l = k/2$). If none of the possible classes receives at least l votes, it is computed the average distance between the test sample and its NCNs from a same class. In such a case, the given pattern will be finally assigned to the class with the least average distance among all classes voted.

This alternative is similar to that defined for the k-NN rule with a reject option [23]. However, it mainly differs from such a classifier in that the reject option is here substituted for an additional distance test.

1. Find the k NCNs of q.
2. If there exists a class with at least $l = k/2$ votes, then assign q to such a class.
3. If not, compute the average distance between q and each one of the classes voted. Assign q to that class with the least average distance.

3.2.2 Edited k-NCN Rule

A new proposal [40] corresponds to a matter of editing the training prototypes because it considers only the subset of NCNs that are correctly labelled according to the NN classifier. After computing the k NCNs of a given test sample, we select those whose NN among all prototypes in the TS has their same class label assigned. The advantage of using an editing strategy comes from the fact that it tends to remove outliers and retain only prototypes grouped into clustered regions with no overlapping among regions from different classes.

This classifier combines the surrounding neighbourhood provided by the plain k-NCN rule with the conventional nearest neighbourhood, here used in the editing stage.

1. Find the k NCNs of q.
2. Select the subset of NCNs, $X^q = \{x_1^q, \ldots, x_j^q\}$, $j \leq k$, correctly labelled according to the NN rule.
3. Assign q to the class with a majority of votes among those j NCNs (resolve ties randomly).

3.2.3 Pairwise k-NCN Rule

Grabowski [19] proposed a modification of the k-NCN classifier, which is analogous to the scheme introduced by Józwik and Vernazza [28] for the conventional k-NN rule. Thus, instead of dealing with a problem of L different classes, it is decomposed into $L \cdot (L-1)$ two-class problems. Then, the given unclassified point is submitted to all component classifiers and the final decision is obtained by simple voting. Such an ensemble of classifiers is called *pairwise k-NCN* or *parallel net of binary k-NCN classifiers*.

Nevertheless, it is to be noted that preliminary experiments conducted by Grabowski [19] suggest that the classification accuracy obtained by the pairwise approach are insignificantly better than those achieved by the plain k-NCN decision rule.

4 Prototype Selection Techniques

As already mentioned in Section 1, prototype selection aims at obtaining a TS in which both computational efficiency and classification accuracy are improved. While editing approaches try to eliminate erroneously labelled prototypes from the original TS and clean possible overlapping among different class regions, condensing schemes reduce the TS set size without an important accuracy degradation. The heuristic nature of condensing algorithms contrasts with the strong statistical foundation of the most popular editing techniques.

The first work on editing corresponds to that of Wilson [49], and then many others has followed [2, 13, 15, 30, 31, 38, 44]. Differences among most approaches refer to the classification rule employed for editing purposes along with the error estimate and the stopping criterion considered [16]. The error estimate is only used in this context to decide which prototypes must be removed from the original TS. The stopping criterion provides the possibility of applying editing in a iterative way.

On the other hand, Hart [21] proposed to minimize the number of training prototypes by by selectively discarding the redundant part of the TS. The basic idea is that prototypes in the TS may be very similar and some do not add extra information and thus may be eliminated. In brief, Hart's scheme consists of discarding those prototypes from the TS that are not necessary to correctly classify the rest of training patterns.

This section provides several editing and condensing algorithms based on some kind of surrounding neighbourhood. More specifically, proximity graphs have been applied both to editing and condensing [38, 45], and the nearest centroid neighbourhood has been used in many different editing procedures [15, 41]. Furthermore, an algorithm involving a combined application of editing and condensing both based on proximity graphs is also presented [38]. This combined technique includes a fast recomputation of the corresponding graph structure in a very simple way.

4.1 Editing Through Proximity Graphs

Most of classical editing algorithms consist of discarding prototypes with a sufficiently heterogeneous neighbourhood in terms of class labels. Thus it seems obvious that proximity graphs can be used as well to obtain an appropriate edited set of prototypes.

The graph-based approach [38] consists of applying the general idea of Wilson's editing algorithm [49] but using the GN or RN of each prototype, instead of the classical nearest neighborhood, in order to estimate mislabelled patterns. In a few words, the simplest graph editing can be summarised as follows: after computing the graph neighborhood of each training prototype, discard those patterns that are misclassified by their graph neighbors (instead of their k-NN). In practice, a first approach to this general editing scheme can be written as follows:

1. Construct the proximity graph corresponding to the original TS, say X.
2. For each $x_i \in X$ do:
   - Discard x_i if there is a majority of graph neighbors from a different class.

A further refinement of this general idea consists of taking not only the graph neighbours of a point, but also the neighbours of the graph neighbours from its same class (i.e., a subset of some of the second level graph neighbours). Actually, we are trying to add reliability to the detection of outliers

close to the boundaries among classes. Therefore, the previous algorithm can be modified as follows:

1. Construct the proximity graph corresponding to the original TS, say X.
2. For each $x_i \in X$ misclassified by its graph neighbours do:
   - Consider the subgraph, S, given by x_i and all its graph neighbours from its same class.
   - Discard x_i if the graph neighbourhood of S has a majority of neighbours from some different class than x_i

4.2 Toussaint's Condensing

Toussaint et al. [45] proposed a condensing algorithm based on the employment of the GG or RNG of the TS. It is inspired from the *Voronoi condensing algorithm*, which finds a reduced TS by using the Voronoi diagram of the TS. The Voronoi condensing scheme is the only procedure that reduces the TS in such a way that the NN boundaries, defined by the reduced set and the original TS are exactly the same, i.e., the reduced set is decision boundary consistent and therefore, also TS consistent.

Nevertheless, the Voronoi condensing algorithm requires the construction of the Voronoi diagram, which is still a time consuming process [37]. Any algorithm in the worst case will take at least $O(n^{d/2})$ time. This practical drawback leads to consider alternative approximate methods by using the GG and RNG. Algorithmically, the GG or RNG condensing scheme can be written as follows:

1. Construct the proximity graph corresponding to the original TS, say X.
2. Visit each node of the proximity graph and mark the visited node if all its graph neighbours are not from the same class as the node visited.
3. Discard all prototypes in X corresponding to nodes in the proximity graph that are not marked.

4.3 A Combined Editing-Condensing Algorithm

The algorithms just introduced can be extended in a number of ways. One of the most obvious consists of obtaining an edited-condensed set using

the same graph structure. In fact, editing and condensing are two closely related and complementary techniques [13]: condensing makes sense only when the classes are clustered and well-separated, which constitutes the main objective of any editing algorithm.

There exist some practical advantages if we are to apply both editing and condensing using graph neighbourhood-based approaches. In particular, computation can be saved if part of the proximity information utilised for editing can be reused for condensing. Sánchez et al. [38] presented a simple way to apply graph editing-condensing while keeping the computational burden very close to the cost of computing the proximity graph only.

To employ Toussaint's condensing after the graph editing algorithms using the same graph structure, the edges must be recomputed when the discarded prototypes along with all their edges are removed from the graph. After this trivial step, it may be necessary to add new edges between pairs of non-neighbour nodes. From the definition of graph neighbours, we can conclude that only pairs of nodes whose zone of influence (that is, diameter hypersphere or lune) held an eliminated node can have a new edge. Otherwise, the reason that made a pair of points non-neighbours still holds after editing.

Bearing this in mind, it is possible to store for each pair of non-neighbour nodes the first detected node inside its zone of influence when the graph was computed for the first time. This makes selection of new edges much faster because only the (few) pairs of nodes whose marked prototype has been discarded during editing need to perform a search through the whole set. Note that the nodes are arranged in random order and therefore, the "first detected node" can be any node. Thus, the algorithm can be written as follows:

1. Construct the proximity graph, $G = (V, E)$, corresponding to the original TS X and, for each pair of points $(x_i, x_j) \in E$, mark the first node that lies inside its zone of influence.
2. Graph-based editing.
3. Construct the subgraph, $G' = (V', E')$, corresponding to non-discarded nodes.
4. For each pair x_i, x_j from V', $(x_i, x_j) \notin E'$ and whose stored node is not in V', put an edge if no other node from V' lies inside its zone of influence.
5. Graph-based condensing with the recomputed graph.

It is worth pointing out that modification in Step 1 requires no extra time, while the Step 4 is expected to be applied to a very reduced number of edges, as confirmed by experimental tests [38].

4.4 k-NCN Editing Strategies

An editing algorithm was proposed in [15] as a way of improving the behaviour of classical editing in the finite sample size case. This technique corresponds to a slight modification of the original work of Wilson and basically consists of using the leaving-one-out error estimate with the k-NCN classification rule [39].

From a practical point of view, the k-NCN classifier is thought to obtain a more accurate information about prototypes in the TS and more specially, for those being close to decision boundaries. In general, when applied to editing, this results in a practical improvement of the corresponding procedure [15]. The simplest k-NCN editing algorithm can be expressed as follows:

1. Let $S = X$.
2. For each x_i in X do:
   - Discard x_i from S if it is misclassified using the k-NCN rule with prototypes in $X - \{x_i\}$.

4.4.1 Fusion of NN and k-NCN classifiers for editing

An extension to this editing procedure corresponds to that proposed by Sánchez et al. [41]. It combines information about proximity with that about the spatial distribution of prototypes in the TS. This kind of classifier fusion tries to guarantee that the prototypes used for editing are correctly labelled according to the NN rule. Although, in general, it is not possible to remove only prototypes lying in wrong class regions, this technique can be seen as a way of reducing the number of "correct" prototypes discarded during the editing process.

In a few words, after searching for the k NCNs of a training prototype, only those whose NN belongs to their same class will be considered for editing. Therefore, this can be understood as a matter of combining two editings: first, editing the k NCNs by using the NN rule and second, editing a prototype by means of the $j \leq k$ NCNs not discarded in the previous stage. This idea can be algorithmically summarised as follows:

1. Let $S = X$.
2. For each x_i in X do:
   - Identify the k NCNs of x_i in $X - \{x_i\}$.
   - Select the $j \leq k$ NCNs correctly labelled according to the NN rule.
   - Discard x_i from S if it is misclassified using the j-NCN rule with prototypes in $X - \{x_i\}$.

Alternatively, one might use the general k-NN decision rule (instead of the particular 1-NN) for editing the set of the NCNs (Step 2.b of the algorithm). On the other hand, although this scheme is based on the leaving-one-out estimate, other error estimates as holdout and B-fold cross-validation are also directly applicable.

4.4.2 Iterative k-NCN Editing

This second alternative basically consists of editing repeatedly the already edited TS until no more prototypes are removed. The idea of this moderate extension to the k-NCN algorithm is that, if a single application of editing can generally improve the classification accuracy, it is expected that successive deletions of outliers can mean an additional increase in performance [44]. Thus the iterative editing procedure can be written in the following way:

1. Let $S = \emptyset$.
2. While $S \neq X$
   - Let $S = X$.
   - Assign to X the result of applying the plain k-NCN editing to S.

An exhaustive empirical analysis carried out by Sánchez et al. [42] shows that this iterative procedure outperforms the simple k-NCN editing for all values of the parameter k. In their work, they also conclude that the highest differences are found with small values of k. On the other hand, from those experiments, there is enough evidence to state that the iterative procedure "cleans" better the overlap between different class regions than the plain k-NCN editing and as a by-product, the NN decision boundaries approximated by the iterative scheme are much simpler than those of the plain algorithm. Consequently, in practice, the computational burden to classify new input

patterns will be lower when using the iterative edited set than in the case of employing the NCN edited set.

5 Summary

Neighbourhood or similarity based classification techniques have been applied to many real-world environments. To classify a given pattern, these schemes firstly analyse the degree of similarity between it and a subset of prototypes from the TS and then, assign it to the class represented by the most similar examples. The k-NN decision rule constitutes one of the most widely used and simplest approaches to such non-parametric classifiers. Nevertheless, these methods suffer from some important drawbacks that make difficult their application in practice: large memory and time requirements, sensitivity to noisy and mislabelled prototypes, and also to irrelevant features.

Traditionally, neighbourhood classification research has been devoted to the development and analysis of algorithms for overcoming the different NN drawbacks. In recent years, however, several researchers [19, 38, 45, 51] have focused on finding alternative neighbourhood specifications. Surrounding neighbourhood is an interesting example of such similarity definitions, in which not only proximity but also symmetry of prototypes must be analysed.

Surrounding neighbourhood can be obtained from a number of realizations. In this chapter, the nearest centroid neighbourhood, the graph neighbourhood, and the Zhang's neighbourhood have been introduced. All these have been employed in very distinct classification and prototype selection algorithms. Empirical studies have shown that in general, this kind of neighbourhood outperforms the conventional representation given by the nearest neighbourhood.

There are still many issues that could be investigated in the framework of alternative neighbourhood representations for pattern classification. In particular, one of the most important drawbacks related with the approaches introduced in the present chapter refers to the fact that the computations needed to calculate the surrounding neighbourhood of a pattern from the prototypes stored in a large TS can be prohibitively expensive, specially in applications where classification time is important. Thus it seems necessary to propose efficient algorithms for implementing such neighbourhood realizations, or even to define new specifications.

As an immediate challenge, the application of surrounding neighbour-

hood algorithms to other pattern recognition and machine learning domains should be done. More specifically, our present work is primarily addressed to investigate the potential of these methods when applied to problems where one class is much more represented than the others in the TS. It has been observed that this situation, which arises in several practical situations, may produce an important deterioration of the classification accuracy, specially with patterns belonging to the less represented class.

Acknowledgements

This work was partially supported by the grant TIC2000-C03-03 from the Spanish CICYT (Department of Science and Technology).

References

[1] A. Akkus and H.A. Güvenir, k Nearest neighbor classification on feature projections, in L. Saitta (ed.) *Proc. of the 13th International Conference on Machine Learning*, (Morgan Kaufmann, Bari, Italy, 1996) pp. 12-19.

[2] R. Barandela and E. Gasca, Decontamination of training samples for supervised pattern recognition methods. in F.J. Ferri et al. (eds.) *Advances in Pattern Recognition*, (Springer Verlag, Alicante, Spain, 2000) pp. 621-630.

[3] S.O. Belkasim, M. Shridhar, and M. Ahmadi, Pattern classification using an efficient KNNR, *Pattern Recognition* Vol.25, (1992) pp. 1269-1274.

[4] A.L. Blum and P. Langley, Selection of relevant features and examples in machine learning, *Artificial Intelligence* Vol.97, (1997) pp. 245-271.

[5] C.E. Brodley and M.A. Friedl, Identifying mislabeled training data, *Journal of Artificial Intelligence Research* Vol.11, (1999) pp. 131-167.

[6] B.B. Chaudhuri, A new definition of neighborhood of a point in multi-dimensional space, *Pattern Recognition Letters* Vol.17, (1996) pp. 11-17.

[7] S. Cost and S. Salzberg, A weighted nearest neighbor algorithm for learning with symbolic features, *Machine Learning* Vol.10, (1993) pp. 57-78.

[8] T.M. Cover and P.E. Hart, Nearest neighbor pattern classification, *IEEE Trans. on Information Theory* Vol.13, (1967) pp. 21-27.

[9] B.V. Dasarathy, *Nearest Neighbor Norms: NN Pattern Classification techniques*, (IEEE Computer Society Press, Los Alamos, CA, 1991).

[10] B.V. Dasarathy, Minimal consistent subset (MCS) identification for optimal nearest neighbor decision systems design, *IEEE Trans. on Systems, Man, and Cybernetics* Vol.24, (1994), pp. 511-517.

[11] B.V. Dasarathy, J.S. Sánchez, and S. Townsend, Nearest neighbor editing and condensing tools - synergy exploitation, *Pattern Analysis and Applications* Vol.3, (2000) pp. 19-30.

[12] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications*, (Springer-Verlag, Berlin, 1997).

[13] P.A. Devijver and J. Kittler, *Pattern Recognition: A Statistical Approach*, (Prentice Hall, Englewood Cliffs, NJ, 1982).

[14] A. Djouadi and E. Bouktache, A fast algorithm for the nearest-neighbor classifier, *IEEE Trans. on Pattern Analysis and Machine Intelligence* Vol.19, (1997) pp. 277-282.

[15] F.J. Ferri, J.S. Sánchez, and F. Pla, Editing prototypes in the finite sample size case using alternative neighbourhoods, in A. Amin et al. (eds.) *Advances in Pattern Recognition*, (Springer-Verlag, Sydney, Australia, 1998) pp 620-629.

[16] F.J. Ferri, J.V. Albert, and E. Vidal, Considerations about sample-size sensitivity of a family of edited nearest-neighbor rules, *IEEE Trans. on Systems, Man, and Cybernetics-Part B: Cybernetics* Vol.29, (1999) pp. 667-672.

[17] E. Fix and J.L. Hodges, Discriminatory analysis, non-parametric discrimination, consistency properties, *Project 21-49-004, Report No. 4, Contract AF41(128)-3*, (USAF School of Aviation Medicine, Randolph Field, TX, 1951).

[18] G.W. Gates, The reduced nearest neighbor rule, *IEEE Trans. on Information Theory* Vol.18, (1972) pp. 431-433.

[19] S. Grabowski, Experiments with the k-NCN decision rule, in *Proc. of the IX Konferencja Sieci i Systemy Informatyczne*, (Lodz, Poland, 2001) *(in press)*.

[20] D.J. Hand, J.N. Kok, and M.R. Berthold, *Advances in Intelligent Data Analysis*, (Springer Verlag, Berlin, 1999).

[21] P.E. Hart, The condensed nearest neighbor rule, *IEEE Trans. on Information Theory* Vol.14, (1968) pp. 515-516.

[22] K. Hattori and M. Takahashi, A new edited k-nearest neighbor rule in the pattern classification problem, *Pattern Recognition* Vol.33, (2000) pp. 521-528.

[23] M.E. Hellman, The nearest neighbor classification rule with a reject option, *IEEE Trans. on Systems, Science, and Cybernetics* Vol.6, (1970) pp. 179-185.

[24] A.K. Jain *Fundamentals of Digital Image Processing*, (Prentice Hall, Englewood Cliffs, NJ, 1989).

[25] A. Jain and D. Zongker, Feature selection: evaluation, application and small sample performance, *IEEE Trans. on Pattern Analysis and Machine Intelligence* Vol.19, (1997) pp. 153-158.

[26] J.W. Jaromczyk and G.T. Toussaint, Relative neighborhood graphs and their relatives, *Proc. IEEE* Vol.80, (1992) pp. 1502-1517.

[27] G.H. John, R. Kohavi, and K. Pfleger, Irrelevant features and the subset selection problem, in W.W. Cohen and H. Hirsh (eds.) *Proc. of the 11th International Conference on Machine Learning*, (Morgan Kaufmann, New Brunswick, NJ, 1994) pp. 121-129.

[28] A. Józwik and G. Vernazza, Recognition of leucocytes by a parallel k-NN classifier, in *Lecture Notes of ICB Seminar*, (Warsaw, Poland, 1988) pp. 138-153.

[29] R. Kohavi and G.H. John, Wrappers for feature subset selection, *Artificial Intelligence* Vol.97, (1997) pp.273-324.

[30] J. Koplowitz, and T.A. Brown, On the relation of performance to editing in nearest neighbor rules., *Pattern Recognition* Vol.13, (1981) pp. 251-255.

[31] L.I. Kuncheva, Editing for the k-nearest neighbors rule by a genetic algorithm, *Pattern Recognition Letters* Vol.16, (1995) pp. 809-814.

[32] L.I. Kuncheva and L.C. Jair, Nearest neighbor classifier: simultaneous editing and feature selection, *Pattern Recognition Letters* Vol.20, (1999) pp. 1149-1156.

[33] P. Langley and W. Iba, Average-case analysis of a nearest neighbor algorithm, in R. Bajcsy (ed.) *Proc. of the 13th International Joint Conference on Artificial Intelligence*, (Morgan Kaufmann, Chambéry, France, 1993) pp. 889-894.

[34] C.X. Ling and H. Wang, Computing optimal attribute weight settings for nearest neighbor algorithms, *Artificial Intelligence Review* Vol.11, (1997) pp. 255-272.

[35] F. Moreno-Seco, J.M. Iñesta, L. Micó, and J. Oncina, Fast k-neighbour classification of human vertebrae levels, in J.S. Sánchez and F. Pla (ed.), *Pattern Recognition and Image Analysis II*, (Publicacions de la Universitat Jaume I, Castellón, Spain, 2001) pp. 343-348.

[36] J.F. O'Callaghan, An alternative definition for neighborhood of a point, *IEEE Trans. on Computers* Vol.24, (1975) pp. 1121-1125.

[37] F.P. Preparata and M.I. Shamos, *Computational Geometry. An introduction*, (Springer, New York, 1985).

[38] J.S. Sánchez, F. Pla, and F.J. Ferri, Prototype selection for the nearest neighbour rule through proximity graphs, *Pattern Recognition Letters* Vol.18, (1997) pp. 507-513.

[39] J.S. Sánchez, F. Pla, and F.J. Ferri, On the use of neighbourhood-based non-parametric classifiers, *Pattern Recognition Letters* Vol.18, (1997) pp. 1179-1186.

[40] J.S. Sánchez, F. Pla, and F.J. Ferri, Improving the k-NCN classification rule through heuristic modifications, *Pattern Recognition Letters* Vol. 19, (1998) pp. 1165-1170.

[41] J.S. Sánchez, R. Barandela, R. Alejo, and A.I. Marqués, Performance evaluation of prototype selection algorithms for nearest neighbor classification, in D.L. Borges and S.-T. Wu (eds.) *Proc. of the 14th. Brazilian Symposium on Computer Graphics and Image Processing*, (IEEE Computer Society Press, Florianópolis, Brazil, 2001) pp. 44-50.

[42] J.S. Sánchez, R. Barandela, A.I. Marqués, R. Alejo, and J. Badenas, Analysis of new techniques to obtain quality training sets, *Pattern Recognition Letters*, (2001) *(in press)*.

[43] W. Siedlecki and J. Sklansky, On automatic feature selection, *International Journal of Pattern Recognition and Artificial Intelligence* Vol.2, (1988) pp. 197-220.

[44] I. Tomek, An experiment with the edited nearest neighbor rule, *IEEE Trans. on Systems, Man, and Cybernetics* Vol.6, (1976) pp. 448-452.

[45] G.T. Toussaint, B.K. Bhattacharya, and R.S. Poulsen, The application of Voronoi diagrams to nonparametric decision rules, in L. Billard (ed.) *Proc. of the 16th Symposium on the Interface: Computer Science and Statistics*, (Atlanta, GA, 1984) pp. 97-108.

[46] J.I. Toriwaki and S. Yokoi, Voronoi and related neighbors on digitized two dimensional space with applications to texture analysis, in G.T. Toussaint (ed.) *Computational Morphology*, (North-Holland, 1988) pp. 207-228.

[47] M. Tüceryan and T. Chorzempa, Relative sensitivity of a family of closest-point graphs in computer vision applications, *Pattern Recognition* Vol.24, (1991) pp. 361-373.

[48] E. Vidal, An algorithm for finding nearest neighbours in (approximately) constant average time, *Pattern Recognition Letters* Vol.4, (1986) pp. 145-147.

[49] D.L. Wilson, Asymptotic properties of nearest neighbor rules using edited data, *IEEE Trans. on Systems, Man, and Cybernetics* Vol.2, (1972) pp. 408-421.

[50] T. Young and K.-S. Fu, *Handbook of Pattern Recognition and Image Processing*, (Academic Press, Orlando, FL, 1986).

[51] J. Zhang, Y. Yim, and J. Yang, Intelligent selection of instances for prediction functions in lazy learning algorithms, *Artificial Intelligence Review* Vol.11, (1997) pp. 175-191.