Using the Dual of Proximity Graphs for Binary Decision Tree Design

J.S. Sánchez, F. Pla, and M.C. Herrero

Departament d'Informàtica, Universitat Jaume I.
E-12071 Castelló, Spain.
{sanchez,pla,cherrero}@uji.es, +34 964 728 350

Abstract. This paper describes an algorithm to design a tree-structured classifier with the hyperplanes associated with a set of prototypes. The main purpose of this technique consists of defining a classification scheme whose result is close to that produced by the Nearest Neighbour decision rule, but getting important computation savings during classification.

Keywords: Nearest Neighbour; Decision Tree; Classification; Gabriel Graph; Relative Neighbourhood Graph

1 Introduction

Non-parametric classification by means of a distance measure is one of the earliest methods used in Pattern Recognition. The Nearest Neighbour (NN) rule [6] is an appropriate example of this kind of classifiers. Given a set of n previously labelled prototypes (namely, training set) in a d-dimensional feature space, this rule assigns to a given sample the same class than the closest prototype in the training set.

Despite of the simplicity and effectiveness of the NN classifiers, it is well known that these schemes suffer from some practical drawbacks such as needing a lot of memory and computational resources for large training sets. This is why numerous investigations have been carried out on this technique in order to find the nearest neighbour of an unknown test sample with as few computations as possible. For example, several strategies have been proposed to devise fast algorithms to search for the nearest neighbour [1,8,10,11,23]. On the other hand, condensing methods [5,7,12,19,22] have been directed to reduce the training set size by selecting some prototypes among the available ones. Nevertheless, it is worth mentioning that these techniques cannot avoid a certain degradation of the classification performance.

A third possibility consists of generating new samples instead of selecting a subset of prototypes [4,15,24]. This approach corresponds to a kind of supervised vector quantization, usually referred to as LVQ, where a discrimination purpose is taken into account.

Finally, other alternatives focus on the use of some data structures which allow a more efficient classification than computing distances from a given test

F.J. Ferri et al. (Eds.): SSPR&SPR 2000, LNCS 1876, pp. 482–490, 2000.
© Springer-Verlag Berlin Heidelberg 2000

sample to all prototypes in the training set. Most of these approaches are based on a certain partition of the d-dimensional feature space. In particular, kd tree methods [9,11] are a popular tool to obtain a tree-like classifier from a decomposition of the feature space with a set of hyperplanes. A similar solution [2,16] is concerned to an implementation of the NN rule in the form of a neural network from the Voronoi Diagram (VD) associated with a set of points.

Related to the latter group of the aforementioned proposals, it has recently been introduced an algorithm to design a decision tree equivalent to the NN rule from partitioning the feature space by means of a subset of the hyperplanes that define the VD associated with the training set [21]. Nevertheless, this technique has an important flaw: the use of VD. While VD can be efficiently constructed in 2 dimensions, there are no efficient algorithms for higher dimensional spaces. Obviously, this severely limits the applicability of such an approach.

This paper presents an alternative scheme for designing a decision tree whose classification result will approximate to that of the NN rule. The aim is to overcome the drawbacks related to the construction of VD, which makes infeasible its application to real problems. Thus, it is here proposed to use a geometrical structure defined from a set of proximity graphs, whose construction is computationally cheaper than that of the VD.

The organization of this paper is as follows. Section 2 introduces the main concepts related to decision trees. Section 3 provides a brief overview of the proximity graphs used in this work. The algorithm to derive a decision tree-structured NN classifier from VD is outlined in Section 4. The scheme proposed in this paper is given in Section 5. Finally, the main contributions of this paper along with concluding remarks are summarized in Section 6.

2 Decision Trees

A decision tree (DT) [3] is a particularly useful tool for complex classification tasks because they are performed by a sequence of simple, easy-to-understand tests whose semantics is intuitively clear to domain experts. Using the terminology of books on data structures, the top node of a DT is called the root. In a binary DT, each node has either no child (in such a case, it is called terminal node or leaf, which is associated with a class), or a left child and a right child. Each node is the root of a tree itself. The trees rooted at the children of a node are called the left and right subtrees of that node. The depth of a node is defined as the length of the path from the node to the root. The height of a tree is the maximum depth of any node.

A DT classifier uses some decision functions at non-terminal nodes to determine the class membership of an unknown sample. The evaluation of these decision rules is organized in such a way that the outcome of successive decision functions reduces uncertainty about the sample being considered for classification. Many variants of DT algorithms have been introduced in the literature. Much of this work has concentrated on DTs in which each non-terminal node checks the value of a single feature or attribute [3,18]. When the attributes are

numeric, the tests have the form $x_i > t$, where x_i is one of the attributes of a sample and t is a constant. This kind of DTs are generally called axis-parallel, because the decision functions at each node are equivalent to axis-parallel hyperplanes in the feature space.

Another option is concerned to DTs that test a linear combination of the attributes at each non-terminal node. More precisely, let a prototype take the form $x = x_1, x_2, ..., x_d, C_j$ where C_j is a class label and the x_i's are real-valued attributes. Thus, the decision function at each node will have the form

$$\sum_{i=1}^{d} a_i x_i + a_{d+1} > 0,$$

where $a_1, ..., a_{d+1}$ are real-valued coefficients. This kind of DTs are usually called oblique decision trees because those linear tests are equivalent to hyperplanes at an oblique orientation to the axes [3,13]. In the case of oblique DTs, decomposition of the feature space is in the form of polyhedral partitionings. It seems clear enough that, in many domains, axis-parallel methods will have to approximate the correct model with a staircase-like structure, while an oblique tree-building technique could capture it with a more accurate DT.

Classification by means of a binary DT has got some attractive advantages. First, it establishes a decision technique that is easy to analyze and understand. Second, it uses a systematic way for calculating the class to assign to a given sample. On the other hand, DT is much faster in terms of computing time than a traditional NN approach since it is not required to compute distances from a test sample to all prototypes in the training set, which obviously constitutes an important benefit from a practical point of view (in fact, it just needs to do m comparisons in a tree of height m).

3 Proximity Graphs

Let $X = \{x_1, \ldots, x_n\}$ be a set of n points in R^d, where d denotes the dimensionality of the feature space. Then 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 neighbourhood relation. In such a case, we say that x_i and x_j are *graph neighbours*. 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), the *Relative Neighbourhood Graph* (RNG) [14] and the *Delaunay Triangulation* (DTR) [17] are the most prominent examples of proximity graphs.

Let $d(\cdot,\cdot)$ be the Euclidean distance in R^d. The edges in a GG are 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*. In other words, two points are Gabriel neighbours if and only if there is no other point from X laying in a hypersphere (namely, *hypersphere of influence*) centered at their middle point and whose diameter is the distance between them.

Analogously, the set of edges in the RNG is obtained as follows:

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

Its corresponding geometric interpretation is based on the concept of *lune*, 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. Thus, two points are *relative neighbours* if and only if their lune does not contain other points from X.

Finally, the DTR of a set of points $X = \{x_1, \ldots, x_n\}$ is defined as the dual graph of the VD of the set X, which is a decomposition of R^d into n cells. 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 DTR of the points. Therefore, two points in a DTR are connected with an edge if the boundaries of their Voronoi cells intersect.

As an important property, the DTR is a supergraph of the GG and this one constitutes a supergraph of the RNG. They form a hierarchy of graphs as given bellow:

$$RNG \subseteq GG \subseteq DTR$$

In a similar way to the DTR and the VD associated with a set of points, it is also possible to define a dual structure for the GG and the RNG. Thus, the dual of a GG (or RNG) will be a sequence of convex regions covering the d-dimensional feature space. These regions are obtained by the perpendicular bisector of two points connected by an edge in the corresponding graph structure.

4 Decision Tree Induction for NN Classification

As previously mentioned, several methods try to improve the efficiency of the NN rule by using some kind of fast data structures, which can be induced from a partitioning of the feature space. Most of these approaches use either a DT or

a neural network configuration in order to represent the decomposition of the feature space into regions, each one assigned to a problem class. In fact, different alternatives essentially differ in the structure used as well as the partitioning criterion.

This section focuses on the construction of a DT by means of a subset of the hyperplanes which define the VD associated with a set of prototypes [21]. As a pre-processing step, the VD associated with the training set must be generated. Then, the information necessary to design the DT is derived from the hyperplanes corresponding to the Voronoi boundaries (that is, those hyperplanes separating two Voronoi cells which belong to different classes) related to the training set. Construction of the tree structure is done recursively.

In a few words, the design approach divides the feature space into a number of convex regions such that the populations corresponding to each one become class homogeneous (that is, all prototypes belonging to one region are from the same class). The decision functions for non-terminal nodes of the DT correspond to the hyperplanes of the Voronoi boundaries, while leaves have the class labels assigned to each one of those resulting convex regions.

From a practical point of view, there currently exists an important drawback. When the number of hyperplanes that define the Voronoi boundaries associated with a set of prototypes is large, the resulting DT can grow excessively. Nevertheless, note that this can be yet much more emphasized when used all the hyperplanes of the entire VD derived from the training set, like in the case of the neural network design algorithm [2]. Furthermore, it would be possible to reduce the DT size by previously applying some prototype selection procedure [7] over the original training set.

Nevertheless, the most important limitation of this method is due to the heavy computational loads required for constructing the VD. Consequently, it is to be admitted that the use of this technique is for all practical purposes of little value.

5 A Decision Tree Approximation to NN Classification

The approach proposed in this paper belongs to those methods which construct some kind of structure for efficient NN classification from partitioning the feature space. The design algorithm provided here requires a pre-processing step in which the dual of the GG (or RNG) associated with the set of prototypes is computed. The basis of our alternative is focused on the idea that only those hyperplanes separating two graph cells which belong to different classes (from now on, called hyperplanes from the graph boundaries, or simply splits) are necessary for an accurate decomposition of the feature space. In fact, this idea is according to condensing methods (in particular, to Toussaint's condensing [22]). Thus, the number of hyperplanes used to construct the classifier could be considerably reduced.

The algorithm proposed here follows a top-down methodology to construct a particular DT. It basically consists of a splitting stage to generate non-terminal

nodes, a stopping criterion, and a procedure for assigning a class label to each resulting leaf. Thus, the design approach divides the feature space into a number of convex regions such that the populations corresponding to each one become class homogeneous (that is, all prototypes belonging to one region are from the same class). These regions are derived from the hyperplanes corresponding to the already defined graph boundaries related to the training set. This process is straightforward and gives rise to a DT whose classification result is approximately the same as that of the NN rule.

Let X denote a set of n prototypes belonging to L distinct classes in a d-dimensional feature space and, let $H = \{H_1, H_2, ..., H_m\}$ be the set of the m hyperplanes which define the graph boundaries associated with X. Next, the design procedure will be discussed in detail.

5.1 Splitting

The algorithm begins considering the entire d-dimensional feature space as a unique convex region, R_1. Taking into account that any hyperplane H_i can divide the space into two half-spaces (namely, the positive half-space H_i^+ and the negative one H_i^-), the first hyperplane, $H_1 \in H$, will divide R_1 into two partial convex regions. In such a way, H_1 is designated as the decision rule for the root node.

Further, the procedure goes on taking the next hyperplanes in the set H. At each iteration, the split rule will consist of testing whether the given hyperplane divides some of those partial convex regions R_j, that is, those regions defined from the decision rules (or hyperplanes) corresponding to the nodes in each path of the current DT, or not. When a hyperplane H_i results in a new partition over any of those regions R_j, the DT will be expanded with a new non-terminal node, whose decision rule will correspond to the hyperplane H_i just tested, i.e., the convex region R_j is divided into two new smaller convex regions. This process is iterated by using all the hyperplanes of the graph boundaries $H_i \in H$.

The problem just introduced here is solved by using the Simplex algorithm for linear programming. In order to test whether a hyperplane $H_i \in H$ may partition a convex region or not, the algorithm formulates a linear program which consists of optimizing an arbitrary linear objective function subject to a set of constraints. These constraints will be the inequalities (i.e., constraints of the form $\geq$ or $\leq$) corresponding to the half-spaces which define the given convex region, and the equation (i.e., a constraint of the form $=$) of that hyperplane H_i. The meaning of such a linear problem is as follows: the hyperplane H_i does not divide a convex region if and only if the corresponding linear program is not feasible.

5.2 Assignment of Class Labels to Leaves

With respect to the procedure for labeling the leaves obtained from the splitting stage, a sample on the inside of each convex region (that is, a path in the DT from the root node to a leaf) is generated, and then its nearest neighbour among

the samples in the input training set is searched. Finally, the class label of its nearest neighbour is assigned to the corresponding convex region.

In order to generate a sample belonging to a certain convex region, the edges of such a region are moved inside an arbitrary distance, say δ, and then any vertex of the resulting region is taken. This vertex will be the optimal vector for a new linear program with an arbitrary objective function subject to a set of constraints (in this case, those corresponding to the inequalities of the half-spaces which define the inside convex region).

5.3 Pruning

The size of the resulting oblique DT can be reduced by means of a simple pruning approach. That is, if both children nodes (leaves) of a non-terminal node have assigned the same class label, it means that the hyperplane associated to such a decision node separates two convex regions which belong to the same class and therefore, it is indeed irrelevant for the discrimination process. Hence, such a non-terminal node is assigned to the class attached to its children nodes and, these two nodes are deleted from the previous DT configuration. Obviously, other pruning alternatives can be applied to this DT design algorithm.

5.4 The Overall Algorithm

After a pre-processing step to compute the dual of the GG (or RNG) of the samples in the training set, as well as to extract those hyperplanes corresponding to the graph boundaries, the approach just described can be formally summarized in the following algorithm:

1. Let R denote the set of convex regions in the tree structure after each iteration. Begin with the entire d-dimensional feature space as the unique region in R.
2. *Splitting.* For each $H_i \in H$, do
   - $R' = \emptyset$
   - For each $R_j \in R$ do
     - If R_j can be linearly separated by H_i:
       A. Designate H_i as decision rule for the non-terminal node represented by R_j.
       B. Make $R_{j+1} = R_j \cap H_i^-$ and $R_{j+2} = R_j \cap H_i^+$ the convex regions that represent the children nodes of R_j.
       C. Delete R_j from R and put R_{j+1}, R_{j+2} in R'.
     - Else put R_j in R'.
   - $R = R'$
3. *Assignment.* For each final convex region $R_j \in R$ do
   - Assign to R_j a class label.
4. *Pruning.* Prune the resulting oblique DT.

6 Conclusions and Extensions

In this paper a technique for the design of a specific kind of binary DT, whose classification result is approximately the same as that achieved by the traditional NN rule, has been introduced. More specifically, this DT is equivalent to the proximity-graph-based non-parametric classification rule. The description of such a classifier and its empirical analysis are widely covered in [20].

The DT induction algorithm is based on the use of the graph boundaries (that is, hyperplanes separating graph cells with different class labels) derived from a set of prototypes and, consists of a recursive process which decomposes the feature space into a number of convex regions, each one finally assigned to a certain problem class. Thus, the decision functions for non-terminal nodes of the DT correspond to the hyperplanes of those graph boundaries (or splits), while leaves have the class labels assigned to each one of those resulting convex regions.

The cost of using a conventional NN approach is high since it requires to calculate distances from a point to all prototypes, while using the DT is expected to be lower: in fact, you only need to compare a point with a hyperplane at most m times, being m the maximum depth of the tree. Moreover, note that this technique may still be applied in conjunction with condensing schemes in order to obtain an even more computationally efficient NN classifier.

Future works include investigation of some stopping criteria that allow to obtain a more efficient DT design algorithm without an important degradation in the classification performance. On the other hand, it is still necessary to compare this approach with existing methods, such as *kd* trees. Finally, it is also interesting to perform a theoretical analysis of the algorithm.

Acknowledgements

This work has been artially supported by grants P1B98-03 from the *Fundació Caixa Castelló-Bancaixa*, GV98-14-134 from the *Generalitat Valenciana* and TIC99-0562 from the *Spanish CICYT*.

References

1. S. O. Belkasim, M. Shridhar and M. Ahmadi, "Pattern classification using an efficient KNNR", *Pattern Recognition* **25**, 1269–1274, 1992.
2. N. K. Bose and A. K. Garga, "Neural network design using Voronoi diagrams", *IEEE Trans. on Neural Networks* **4**, 778–787, 1993.
3. L. Breiman, J. H. Friedman, R. A. Olshen and C. J. Stone, *Classification and regression trees*, Chapman & Hall: New York, 1984.
4. C. L. Chang, "Finding prototypes for nearest neighbor classifiers", *IEEE Trans. on Computers* **23**, 1179–1184, 1974.
5. C. H. Chen and A. Józwik, "A sample set condensation algorithm for the class sensitive artificial neural network", *Pattern Recognition Letters* **17**, 819–823, 1996.

6. T. M. Cover and P. E. Hart, "Nearest neighbor pattern classification", *IEEE Trans. on Information Theory* **13**, 21–27, 1967.
7. P. A. Devijver and J. Kittler, *Pattern recognition: a statistical approach*, Prentice Hall: Englewood Cliffs, NJ, 1982.
8. A. Djouadi and E. Bouktache, "A fast algorithm for the nearest-neighbor classifier", *IEEE Trans. on Pattern Analysis and Machine Intelligence* **19**, 277–282, 1997.
9. J. H. Friedman, J. L. Bentley and R. A. Finkel, "An algorithm for finding best matches in logarithmic expected time", *ACM Trans. on Math. Software* **3**, 209–226, 1977.
10. K. Fukunaga and P. M. Narendra, "A branch and bound algorithm for computing k-nearest neighbors", *IEEE Trans. on Computers* **24**, 750–753, 1975.
11. P. J. Grother, G. T. Candela and J. L. Blue, "Fast implementations of nearest neighbour classifiers", *Pattern Recognition* **30**, 459–465, 1997.
12. P. E. Hart, "The condensed nearest neighbor rule", *IEEE Trans. on Information Theory* **14**, 515–516, 1968.
13. D. Heath, S. Kasif and S. Salzberg, "Learning oblique decision trees", In *Proc. 13th Int. Joint Conf. Artificial Intelligence*, 1202–1207, 1993.
14. J.W. Jaromczyk and G.T. Toussaint, "Relative neighborhood graphs and their relatives", *Proc. IEEE* **80**, 1502–1517, 1992.
15. T. Kohonen, "The self-organizing map", *Proc. IEEE* **9**, 1464–1480, 1990.
16. O. J. Murphy, "Nearest neighbor pattern classification perceptrons", *Proc. IEEE*, **78**, 1595–1598, 1990.
17. F.P. Preparata and M.I. Shamos, *Computational geometry. An introduction*, Springer: New York, 1985.
18. J. R. Quinlan, "Induction of decision trees", *Machine Learning* **1**, 81–106, 1986.
19. G. L. Ritter, H. B. Woodruff, S. R. Lowry and T. L. Isenhour, "An algorithm for a selective nearest neighbour decision rule", *IEEE Trans. on Information Theory* **21**, 665–669, 1975.
20. J.S. Sánchez, F. Pla and F.J. Ferri, "On the use of neighbourhood-based non-parametric classifiers", *Pattern Recognition Letters* **18**, 1179–1186, 1997.
21. J.S. Sánchez, F. Pla and F.J. Ferri, "A Voronoi-diagram-based approach to oblique decision tree induction", In *Proc. 14th Int. Conf. Pattern Recognition* **1**, 542–544, 1998.
22. G. T. Toussaint, B. K. Bhattacharya and R. S. Poulsen, "The application of Voronoi diagrams to nonparametric decision rules", In *Computer Science & Statistics: 16th Symp. Interface*, 97–108, 1984.
23. E. Vidal, "An algorithm for finding nearest neighbours in (approximately) constant average time", *Pattern Recognition Letters* **4**, 145–157, 1986.
24. H. Yan, "Prototype optimization for nearest neighbor classifiers using a two-layer perceptron", *Pattern Recognition* **26**, 317–324, 1993.