Cellular Tree Classifiers

Gérard Biau[1,2] and Luc Devroye[3]

[1] Sorbonne Universités, UPMC Univ Paris 06, France
[2] Institut universitaire de France
[3] McGill University, Canada

Abstract. Suppose that binary classification is done by a tree method in which the leaves of a tree correspond to a partition of d-space. Within a partition, a majority vote is used. Suppose furthermore that this tree must be constructed recursively by implementing just two functions, so that the construction can be carried out in parallel by using "cells": first of all, given input data, a cell must decide whether it will become a leaf or an internal node in the tree. Secondly, if it decides on an internal node, it must decide how to partition the space linearly. Data are then split into two parts and sent downstream to two new independent cells. We discuss the design and properties of such classifiers.

1 Introduction

We explore in this note a new way of dealing with the supervised classification problem, inspired by greedy approaches and the divide-and-conquer philosophy. Our point of view is novel, but has a wide reach in a world in which parallel and distributed computation are important. In the short term, parallelism will take hold in massive data sets and complex systems and, as such, is one of the exciting questions that will be asked to the statistics and machine learning fields.

The general context is that of classification trees, which make decisions by recursively partitioning $\mathbb{R}^d$ into regions, sometimes called cells. In the model we promote, a basic computational unit in classification, a cell, takes as input training data, and makes a decision whether a majority rule should be locally applied. In the negative, the data should be split and each part of the partition should be transmitted to another cell. What is original in our approach is that all cells must use **exactly** the same protocol to make their decision—their function is not altered by external inputs or global parameters. In other words, the decision to split depends only upon the data presented to the cell, independently of the overall edifice. Classifiers designed according to this autonomous principle will be called cellular tree classifiers, or simply cellular classifiers.

Decision tree learning is a method commonly used in data mining (see, e.g., [27]). For example, in CART (Classification and Regression Trees, [5]), splits are made perpendicular to the axes based on the notion of Gini impurity. Splits are performed until all data are isolated. In a second phase, nodes are recombined from the bottom-up in a process called pruning. It is this second process that makes the CART trees non-cellular, as global information is shared to manage

the recombination process. Quinlan's C4.5 [26] also prunes. Others split until all nodes or cells are homogeneous (i.e., have the same class)—the prime example is Quinlan's ID3 [25]. This strategy, while compliant with the cellular framework, leads to non-consistent rules, as we point out in the present paper. In fact, the choice of a good stopping rule for decision trees is very hard—we were not able to find any in the literature that guarantee convergence to the Bayes error.

2 Tree classifiers

In the design of classifiers, we have an unknown distribution of a random prototype pair $(\mathbf{X}, Y)$, where $\mathbf{X}$ takes values in $\mathbb{R}^d$ and Y takes only finitely many values, say 0 or 1 for simplicity. Classical pattern recognition deals with predicting the unknown nature Y of the observation $\mathbf{X}$ via a measurable classifier $g : \mathbb{R}^d \to \{0, 1\}$. We make a mistake if $g(\mathbf{X})$ differs from Y, and the probability of error for a particular decision rule g is $L(g) = \mathbb{P}\{g(\mathbf{X}) \neq Y\}$. The Bayes classifier

$$g^\star(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbb{P}\{Y = 1 | \mathbf{X} = \mathbf{x}\} > \mathbb{P}\{Y = 0 | \mathbf{X} = \mathbf{x}\} \\ 0 & \text{otherwise} \end{cases}$$

has the smallest probability of error, that is

$$L^\star = L(g^\star) = \inf_{g:\mathbb{R}^d \to \{0,1\}} \mathbb{P}\{g(\mathbf{X}) \neq Y\}$$

(see, for instance, Theorem 2.1 in [7]). However, most of the time, the distribution of $(\mathbf{X}, Y)$ is unknown, so that the optimal decision $g^\star$ is unknown too. We do not consult an expert to try to reconstruct $g^\star$, but have access to a database $\mathscr{D}_n = (\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)$ of i.i.d. copies of $(\mathbf{X}, Y)$, observed in the past. We assume that $\mathscr{D}_n$ and $(\mathbf{X}, Y)$ are independent. In this context, a classification rule $g_n(\mathbf{x}; \mathscr{D}_n)$ is a Borel measurable function of $\mathbf{x}$ and $\mathscr{D}_n$, and it attempts to estimate Y from $\mathbf{x}$ and $\mathscr{D}_n$. For simplicity, we suppress $\mathscr{D}_n$ in the notation and write $g_n(\mathbf{x})$ instead of $g_n(\mathbf{x}; \mathscr{D}_n)$.

The probability of error of a given classifier g_n is the random variable

$$L(g_n) = \mathbb{P}\{g_n(\mathbf{X}) \neq Y | \mathscr{D}_n\},$$

and the rule is consistent if

$$\lim_{n\to\infty} \mathbb{E}L(g_n) = L^\star.$$

It is universally consistent if it is consistent for all possible distributions of $(\mathbf{X}, Y)$. Many popular classifiers are universally consistent. These include several brands of histogram rules, k-nearest neighbor rules, kernel rules, neural networks, and tree classifiers. There are too many references to be cited here, but the monographs by [7] and [15] will provide the reader with a comprehensive introduction to the domain and a literature review.

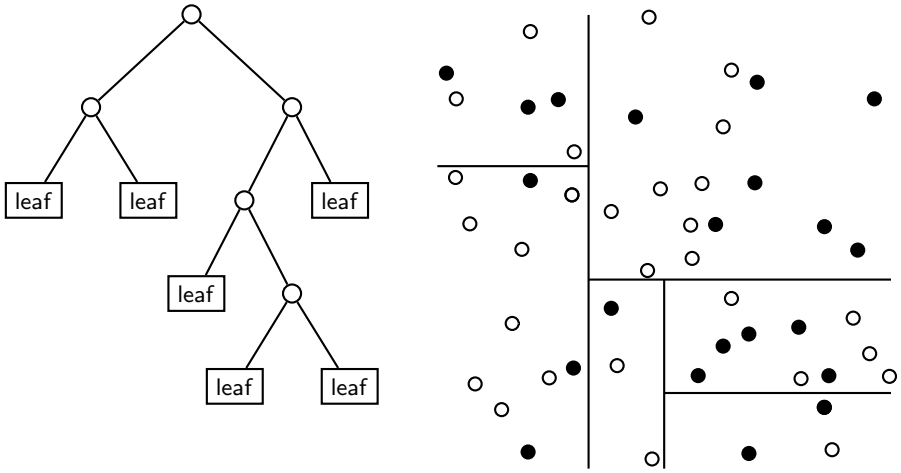

Fig. 1. A binary tree (left) and the corresponding partition (right).

Trees have been suggested as tools for classification for more than thirty years. We mention in particular the early work of Fu [36, 1, 21, 18, 24]. Other references from the 1970s include [20, 3, 23, 30, 34, 12, 8]. Most influential in the classification tree literature was the CART proposal by [5]. While CART proposes partitions by hyperrectangles, linear hyperplanes in general position have also gained in popularity—the early work on that topic is by [19], and [22]. Additional references on tree classification include [14, 2, 16, 17, 35, 33, 31, 6, 9, 10, 32, 13].

3 Cellular trees

In general, classification trees partition $\mathbb{R}^d$ into regions, often hyperrectangles parallel to the axes (an example is depicted in Figure 1). Of interest in this article are binary trees, where each node has exactly 0 or 2 children. If a node u represents the set A and its children u_1, u_2 represent A_1, A_2, then it is required that $A = A_1 \cup A_2$ and $A_1 \cap A_2 = \emptyset$. The root of the tree represents $\mathbb{R}^d$, and the terminal nodes (or leaves), taken together, form a partition of $\mathbb{R}^d$. If a leaf represents region A, then the tree classifier takes the simple form

$$g_n(\mathbf{x}) = \begin{cases} 1 & \text{if } \sum_{i=1}^n \mathbb{1}_{[\mathbf{X}_i \in A, Y_i = 1]} > \sum_{i=1}^n \mathbb{1}_{[\mathbf{X}_i \in A, Y_i = 0]}, \\ 0 & \text{otherwise.} \end{cases} \qquad \mathbf{x} \in A$$

That is, in every leaf region, a majority vote is taken over all $(\mathbf{X}_i, Y_i)$'s with $\mathbf{X}_i$'s in the same region. Ties are broken, by convention, in favor of class 0.

The tree structure is usually data-dependent, and indeed, it is in the construction itself where different trees differ. Thus, there are virtually infinitely many possible strategies to build classification trees. Nevertheless, despite this

great diversity, all tree species end up with two fundamental questions at each node:

① Should the node be split?
② In the affirmative, what are its children?

These two questions are typically answered using **global** information on the tree, such as, for example, a function of the data $\mathscr{D}_n$, the level of the node within the tree, the size of the data set and, more generally, any parameter connected with the structure of the tree. This parameter could be, for example, the total number k of cells in a k-partition tree or the penalty term in the pruning of the CART algorithm (e.g., [5] and [11]).

Our cellular trees proceed from a different philosophy. In short, a cellular tree should, at each node, be able to answer questions ① and ② using **local** information only, without any help from the other nodes. In other words, each cell can perform as many operations as it wishes, provided it uses only the data that are transmitted to it, regardless of the general structure of the tree. Just imagine that the calculations to be carried out at the nodes are sent to different computers, eventually asynchronously, and that the system architecture is so complex that computers do not communicate. Thus, once a computer receives its data, it has to make its own decisions on ① and ② based on this data subset only, independently of the others and without knowing anything of the overall edifice. Once a data set is split, it can be given to another computer for further splitting, since the remaining data points have no influence.

Formally, a cellular binary classification tree is a machine that partitions the space recursively in the following manner. With each node we associate a subset of $\mathbb{R}^d$, starting with $\mathbb{R}^d$ for the root node. We consider binary tree classifiers based on a class $\mathcal{C}$ of possible Borel subsets of $\mathbb{R}^d$ that can be used for splits. A typical example of such a class is the family of all hyperplanes, or the class of all hyperplanes that are perpendicular to one of the axes. Higher order polynomial splitting surfaces can be imagined as well. The class is parametrized by a vector $\sigma \in \mathbb{R}^p$. There is a splitting function $f(\mathbf{x}, \sigma)$, $\mathbf{x} \in \mathbb{R}^d, \sigma \in \mathbb{R}^p$, such that $\mathbb{R}^d$ is partitioned into $A = \{\mathbf{x} \in \mathbb{R}^d : f(\mathbf{x}, \sigma) \geq 0\}$ and $B = \{\mathbf{x} \in \mathbb{R}^d : f(\mathbf{x}, \sigma) < 0\}$. Formally, a cellular split can be viewed as a family of measurable mappings $(\sigma_m)_m$ from $(\mathbb{R}^d \times \{0,1\})^m$ to $\mathbb{R}^p$. In this model, m is the size of the data set transmitted to the cell. Thus, for each possible input size m, we have a map. In addition, there is a family of measurable mappings $(\theta_m)_m$ from $(\mathbb{R}^d \times \{0,1\})^m$ to $\{0,1\}$ that indicate decisions: $\theta_m = 1$ indicates that a split should be applied, while $\theta_m = 0$ corresponds to a decision not to split. In that case, the cell acts as a leaf node in the tree. We note that $(\theta_m)_m$ and $(\sigma_m)_m$ correspond to the decisions given in ① and ②.

Let the set data set be $\mathscr{D}_n$. If $\theta(\mathscr{D}_n) = 0$, the root cell is final, and the space is not split. Otherwise, $\mathbb{R}^d$ is split into

$$A = \left\{\mathbf{x} \in \mathbb{R}^d : f\left(\mathbf{x}, \sigma(\mathscr{D}_n)\right) \geq 0\right\} \quad \text{and} \quad B = \left\{\mathbf{x} \in \mathbb{R}^d : f\left(\mathbf{x}, \sigma(\mathscr{D}_n)\right) < 0\right\}.$$

The data $\mathscr{D}_n$ are partitioned into two groups: the first group contains all $(\mathbf{X}_i, Y_i)$, $i = 1, \ldots, n$, for which $\mathbf{X}_i \in A$, and the second group all others. The groups

are sent to child cells, and the process is repeated. When $\mathbf{x} \in \mathbb{R}^d$ needs to be classified, we first determine the unique leaf set $A(\mathbf{x})$ to which $\mathbf{x}$ belongs, and then take votes among the $\{Y_i : \mathbf{X}_i \in A(\mathbf{x}), i = 1, \ldots, n\}$. Classification proceeds by a majority vote, with the majority deciding the estimate $g_n(\mathbf{x})$. In case of a tie, we set $g_n(\mathbf{x}) = 0$.

A cellular binary tree classifier is said to be randomized if each node in the tree has an independent copy of a uniform $[0, 1]$ random variable associated with it, and θ and σ are mappings that have one extra real-valued component in the input. For example, we could flip an unbiased coin at each node to decide whether $\theta_m = 0$ or $\theta_m = 1$.

4 A consistent cellular tree classifier

At first sight, it appears that there are no universally consistent cellular tree classifiers. Consider for example complete binary trees with k full levels, i.e., there are 2^k leaf regions. We can have consistency when k is allowed to depend upon n. An example is the median tree (see Section 20.3 in [7]). When $d = 1$, split by finding the median element among the $\mathbf{X}_i$'s, so that the child sets have cardinality given by $\lfloor (n-1)/2 \rfloor$ and $\lceil (n-1)/2 \rceil$, where $\lfloor . \rfloor$ and $\lceil . \rceil$ are the floor and ceiling functions. The median itself does stay behind and is not sent down to the subtrees, with an appropriate convention for breaking cell boundaries as well as empty cells. Keep doing this for k rounds—in d dimensions, one can either rotate through the coordinates for median splitting, or randomize by selecting uniformly at random a coordinate to split orthogonally.

This rule is known to be consistent as soon as the marginal distributions of $\mathbf{X}$ are nonatomic, provided $k \to \infty$ and $k2^k/n \to 0$. However, this is not a cellular tree classifier. While we can indeed specify σ_m, it is impossible to define θ_m because θ_m cannot be a function of the global value of n. In other words, if we were to apply median splitting and decide to split for a fixed k, then the leaf nodes would all correspond to a fixed proportion of the data points. It is clear that the decisions in the leaves are off with a fair probability if we have, for example, Y independent of $\mathbf{X}$ and $\mathbb{P}\{Y = 1\} = 1/2$. Thus, we cannot create a cellular tree classifier in this manner.

In view of the preceding discussion, it seems paradoxical that there indeed exist universally consistent cellular tree classifiers. (We note here that we abuse the word "universal"—we will assume throughout, to keep the discussion at a manageable level, that the marginal distributions of $\mathbf{X}$ are nonatomic. But no other conditions on the joint distribution of $(\mathbf{X}, Y)$ are imposed.) Our construction follows the median tree principle and uses randomization. The original work on the solution appears in [4].

From now on, to keep things simple, it is assumed that the marginal distributions of $\mathbf{X}$ are nonatomic. The cellular splitting method σ_m described in this section mimics the median tree classifier discussed above. We first choose a dimension to cut, uniformly at random from the d dimensions, as rotating through the dimensions by level number would violate the cellular condition.

The selected dimension is then split at the data median, just as in the classical median tree. Repeating this for k levels of nodes leads to 2^k leaf regions. On any path of length k to one of the 2^k leaves, we have a deterministic sequence of cardinalities $n_0 = n(\text{root}), n_1, n_2, \ldots, n_k$. We always have $n_i/2 - 1 \leq n_{i+1} \leq n_i/2$. Thus, by induction, one easily shows that, for all i,

$$\frac{n}{2^i} - 2 \leq n_i \leq \frac{n}{2^i}.$$

In particular, each leaf has at least $\max(n/2^k - 2, 0)$ points and at most $n/2^k$. The novelty is in the choice of the decision function. This function ignores the data altogether and uses a randomized decision that is based on the size of the input. More precisely, consider a nonincreasing function $\varphi : \mathbb{N} \to (0, 1]$ with $\varphi(0) = \varphi(1) = 1$. Cells correspond in a natural way to sets of $\mathbb{R}^d$. So, we can and will speak of a cell A, where $A \subset \mathbb{R}^d$. The number of data points in A is denoted by $N(A)$:

$$N(A) = \sum_{i=1}^{n} \mathbb{1}_{[\mathbf{X}_i \in A]}.$$

Then, if U is the uniform $[0, 1]$ random variable associated with the cell A and the input to the cell is $N(A)$, the stopping rule ① takes the form:

① Put $\theta = 0$ if

$$U \leq \varphi\left(N(A)\right).$$

In this manner, we obtain a possibly infinite randomized binary tree classifier. Splitting occurs with probability $1 - \varphi(m)$ on inputs of size m. Note that no attempt is made to split empty sets or singleton sets. For consistency, we need to look at the random leaf region to which $\mathbf{X}$ belongs. This is roughly equivalent to studying the distance from that cell to the root of the tree.

In the sequel, the notation $u_n = \mathrm{o}(v_n)$ (respectively, $u_n = \omega(v_n)$ and $u_n = \mathrm{O}(v_n)$) means that $u_n/v_n \to 0$ (respectively, $v_n/u_n \to 0$ and $u_n \leq Cv_n$ for some constant C) as $n \to \infty$. Many choices $\varphi(m) = \mathrm{o}(1)$, but not all, will do for us. The next lemma makes things more precise.

Lemma 1 *Let $\beta \in (0, 1)$. Define*

$$\varphi(m) = \begin{cases} 1 & \text{if } m < 3 \\ 1/\log^\beta m & \text{if } m \geq 3. \end{cases}$$

Let $K(\mathbf{X})$ denote the random path distance between the cell of $\mathbf{X}$ and the root of the tree. Then

$$\lim_{n\to\infty} \mathbb{P}\left\{K(\mathbf{X}) \geq k_n\right\} = \begin{cases} 0 & \text{if } k_n = \omega(\log^\beta n) \\ 1 & \text{if } k_n = \mathrm{o}(\log^\beta n). \end{cases}$$

Proof. Let us recall that, at level k, each cell of the underlying median tree contains at least $\max(n/2^k - 2, 0)$ points and at most $n/2^k$. Since the function

$\varphi(.)$ is nonincreasing, the first result follows from this:

$$\begin{aligned}\mathbb{P}\{K(\mathbf{X}) \geq k_n\} &\leq \prod_{i=0}^{k_n-1} \left(1-\varphi\left(\lfloor n/2^i \rfloor\right)\right) \\ &\leq \exp\left(-\sum_{i=0}^{k_n-1} \varphi\left(\lfloor n/2^i \rfloor\right)\right) \\ &\leq \exp\left(-k_n\varphi(n)\right).\end{aligned}$$

The second statement follows from

$$\mathbb{P}\{K(\mathbf{X}) < k_n\} \leq \sum_{i=0}^{k_n-1} \varphi\left(\lceil n/2^i - 2 \rceil\right) \leq k_n \varphi\left(\lceil n/2^{k_n} \rceil\right),$$

valid for all n large enough since $n/2^{k_n} \to \infty$ as $n \to \infty$. □

Lemma 1, combined with the median tree consistency result of [7], suffices to establish consistency of the randomized cellular tree classifier.

Theorem 1 *Let β be a real number in $(0,1)$. Define*

$$\varphi(m) = \begin{cases} 1 & \text{if } m < 3 \\ 1/\log^{\beta} m & \text{if } m \geq 3. \end{cases}$$

Let g_n be the associated randomized cellular binary tree classifier. Assume that the marginal distributions of $\mathbf{X}$ are nonatomic. Then the classification rule g_n is consistent:

$$\lim_{n\to\infty} \mathbb{E}L(g_n) = L^{\star}.$$

Proof. By $\mathrm{diam}(A)$ we mean the diameter of the cell A, i.e., the maximal distance between two points of A. We recall a general consistency theorem for partitioning classifiers whose cell design depends on the $\mathbf{X}_i$'s only (see Theorem 6.1 in [7]). According to this theorem, such a classifier is consistent if both

1. $\mathrm{diam}(A(\mathbf{X})) \to 0$ in probability as $n \to \infty$, and
2. $N(A(\mathbf{X})) \to \infty$ in probability as $n \to \infty$,

where $A(\mathbf{X})$ is the cell of the random partition containing $\mathbf{X}$.

Condition 2. is proved in Lemma 1. Notice that

$$\begin{aligned}N\left(A(\mathbf{X})\right) &\geq \frac{n}{2^{K(\mathbf{X})}} - 2 \\ &\geq \mathbb{1}_{[K(\mathbf{X}) < \log^{(\beta+1)/2} n]} \left(\frac{n}{2^{\log^{(\beta+1)/2} n}} - 2\right) \\ &= \omega(1)\mathbb{1}_{[K(\mathbf{X}) < \log^{(\beta+1)/2} n]}.\end{aligned}$$

Therefore, by Lemma 1, $N\left(A(\mathbf{X})\right) \to \infty$ in probability as $n \to \infty$.

To show that $\mathrm{diam}(A(\mathbf{X})) \to 0$ in probability, observe that on a path of length $K(\mathbf{X})$, the number of times the first dimension is cut is binomial $(K(\mathbf{X}), 1/d)$. This tends to infinity in probability. Following the proof of Theorem 20.2 in [7], the diameter of the cell of $\mathbf{X}$ tends to 0 in probability with n. Details are left to the reader. □

Let us finally take care of the randomization. Can one do without randomization? The hint to the solution of that enigma is in the hypothesis that the data elements in $\mathscr{D}_n$ are i.i.d. The median classifier does not use the ordering in the data. Thus, one can use the randomness present in the permutation of the observations, e.g., the ℓ-th components of the $\mathbf{X}_i$'s can form $n!$ permutations if ties do not occur. This corresponds to $(1 + \mathrm{o}(1))n \log_2 n$ independent fair coin flips, which are at our disposal. Each decision to split requires on average at most 2 independent bits. The selection of a random direction to cut requires no more than $1 + \log_2 d$ independent bits. Since the total tree size is, with probability tending to 1, $\mathrm{O}(2^{\log^{\beta+\varepsilon} n})$ for any $\varepsilon > 0$, a fact that follows with a bit of work from summing the expected number of nodes at each level, the total number of bits required to carry out all computations is

$$\mathrm{O}\left((3 + \log_2 d)2^{\log^{\beta+\varepsilon} n}\right),$$

which is orders of magnitude smaller than n provided that $\beta + \varepsilon < 1$. Thus, there is sufficient randomness at hand to do the job. How it is actually implemented is another matter, as there is some inevitable dependence between the data sets that correspond to cells and the data sets that correspond to their children. We will not worry about the finer details of this in the present paper.

References

1. Anderson, A.C. and Fu, K.S.: Design and development of a linear binary tree classifier for leukocytes. Technical Report TR-EE-79-31, Purdue University (1979)
2. Argentiero, P., Chin, R., and Beaudet, P.: An automated approach to the design of decision tree classifiers. IEEE Transactions on Pattern Analysis and Machine Intelligence, 4:51–57 (1982)
3. Bartolucci, L.A., Swain, P.H., and Wu, C.: Selective radiant temperature mapping using a layered classifier. IEEE Transactions on Geosciences and Electronics, 14:101–106 (1976)
4. Biau, G., and Devroye, L.: Cellular tree classifiers. Electronic Journal of Statistics, 7:1875–1912 (2013)
5. Breiman, L., Friedman, J.H., Olshen, R.A., and Stone, C.J.: Classification and Regression Trees. Chapman & Hall, New York (1984)
6. Chou, P.A.: Optimal partitioning for classification and regression trees. IEEE Transactions on Pattern Analysis and Machine Intelligence, 13:340–354 (1991)
7. Devroye, L., Györfi, L., and Lugosi, G.: A Probabilistic Theory of Pattern Recognition. Springer, New York (1996)
8. Friedman, J.H.: A tree-structured approach to nonparametric multiple regression. In T. Gasser and M. Rosenblatt, editors, Smoothing Techniques for Curve Estimation, pages 5–22, Heidelberg, 1979. Lecture Notes in Mathematics #757, Springer

9. Gelfand, S.B. and Delp., E.J.: On tree structured classifiers. In I.K. Sethi and A.K. Jain, editors, Artificial Neural Networks and Statistical Pattern Recognition, Old and New Connections, pages 71–88, Amsterdam (1991). Elsevier Science Publishers
10. Gelfand, S.B., Ravishankar, C.S., and Delp, E.J.: An iterative growing and pruning algorithm for classification tree design. IEEE Transactions on Pattern Analysis and Machine Intelligence, 13:163–174 (1991)
11. Gey, S. and Nédélec, E.: Model selection for CART regression trees. IEEE Transactions on Information Theory, 51:658–670 (2005).
12. Gordon, L. and Olshen, R.A.: Asymptotically efficient solutions to the classification problem. The Annals of Statistics, 6:515–533 (1978)
13. Guo, H. and Gelfand, S.B.: Classification trees with neural network feature extraction. IEEE Transactions on Neural Networks, 3:923–933 (1992)
14. Gustafson, D.E., Gelfand, S., and Mitter, S.K.: A nonparametric multiclass partitioning method for classification. In Proceedings of the Fifth International Conference on Pattern Recognition, pages 654–659 (1980)
15. Györfi, L., Kohler, M., Krzyżak, A., and Walk, H.: A Distribution-Free Theory of Nonparametric Regression. Springer, New York (2002)
16. Hartmann, C.R.P., Varshney, P.K., Mehrotra, K.G., and Gerberich, C.L.: Application of information theory to the construction of efficient decision trees. IEEE Transactions on Information Theory, 28:565–577 (1982)
17. Kurzynski, M.W.: The optimal strategy of a tree classifier. Pattern Recognition, 16:81–87 (1983)
18. Lin, Y.K. and Fu, K.S.: Automatic classification of cervical cells using a binary tree classifier. Pattern Recognition, 16:69–80 (1983)
19. Loh, W.Y. and Vanichsetakul, N.: Tree-structured classification via generalized discriminant analysis. Journal of the American Statistical Association, 83:715–728 (1988)
20. Meisel, W.S. and Michalopoulos, D.A.: A partitioning algorithm with application in pattern classification and the optimization of decision trees. IEEE Transactions on Computers, 22:93–103 (1973)
21. Mui, J.K. and Fu, K.S.: Automated classification of nucleated blood cells using a binary tree classifier. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2:429–443 (1980)
22. Park, Y. and Sklansky, J.: Automated design of linear tree classifiers. Pattern Recognition, 23:1393–1412 (1990)
23. Payne, H.J. and Meisel, W.S.: An algorithm for constructing optimal binary decision trees. IEEE Transactions on Computers, 26:905–916 (1977)
24. Qing-Yun, S. and Fu, K.S.: A method for the design of binary tree classifiers. Pattern Recognition, 16:593–603 (1983)
25. Quinlan, J.R.: Induction of decision trees. Machine Learning, 1:81–106 (1986)
26. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, San Mateo (1993)
27. Rokach, L. and Maimon, O.: Data Mining with Decision Trees: Theory and Applications. World Scientific, Singapore (2008)
28. Samet, H.: The quadtree and related hierarchical data structures. Computing Surveys, 16:187–260 (1984)
29. Samet, H.: The Design and Analysis of Spatial Data Structures. Addison-Wesley, Reading (1990)
30. Sethi, I.K. and Chatterjee, B.: Efficient decision tree design for discrete variable pattern recognition problems. Pattern Recognition, 9:197–206 (1977)

31. Shlien, S.: Multiple binary decision tree classifiers. Pattern Recognition, 23:757–763 (1990)
32. Simon, H.U.: The Vapnik-Chervonenkis dimension of decision trees with bounded rank. Information Processing Letters, 39:137–141 (1991)
33. Suen, C.Y. and Wang, Q.R.: Large tree classifier with heuristic search and global training. IEEE Transactions on Pattern Analysis and Machine Intelligence, 9:91–101 (1987)
34. Swain, P.H. and Hauska, H.: The decision tree classifier: Design and potential. IEEE Transactions on Geosciences and Electronics, 15:142–147 (1977)
35. Wang, Q.R. and Suen, C.Y.: Analysis and design of a decision tree based on entropy reduction and its application to large character set recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 6:406–417 (1984)
36. You, K.C. and Fu, K.S.: An approach to the design of a linear binary tree classifier. In Proceedings of the Symposium of Machine Processing of Remotely Sensed Data, pages 3A–1–10, West Lafayette (1976). Purdue University