STree: A Single Multi-class Oblique Decision Tree Based on Support Vector Machines

Ricardo Montañana(✉), Jose A. Gámez, and Jose M. Puerta

Computing Systems Department, University of Castilla-La Mancha, Albacete, Spain
Ricardo.Montanana@alu.uclm.es, {Jose.Gamez,Jose.Puerta}@uclm.es

Abstract. We propose a new oblique decision tree algorithm based on support vector machines. Our algorithm produces a single model for a multi-class target variable. On the contrary to previous works that manage the multi-class problem by using clustering at each split, we test all the one-vs-rest labels at each split, choosing the one which minimizes an impurity measure. The experimental evaluation carried out over 49 datasets shows that our algorithm is ranked before those used for comparison, and significantly outperforms all of them when the SVM hyperparameters are carefully tuned.

Keywords: Oblique decision trees · Supervised classification · SVM

1 Introduction

Decision trees (DT) [12] are one of the most used classification models in data mining, with recent success stories such as its use by Microsoft Kinect for real-time human pose estimation [19]. This is due, among other advantages, to their high performance and structural simplicity, which make them easily understandable by humans, increasing in this way the confidence to use them in real world applications.

In this work we focus on supervised classification, where the goal is to induce a function or *classifier* $f : X_1 \times \cdots \times X_n \rightarrow Y$, where $X_1, \ldots, X_n$ are the predictive attributes that define the object to be classified and Y is the target variable or *class*, which takes values in a set of finite and disjoint categories (or *labels*), $Y = \{y_1, \ldots, y_k\}$. When a DT is used as classification function (f), it is a tree-shaped data structure where each leaf node is labeled with a value of Y and the inner nodes contain *tests* related to the predictive attributes. Branches coming out from an inner node represent the different answers to its associated test. An object (instance) is classified by following the path from the root to a leaf, using its attribute values to answer the tests at inner nodes. The object is classified with the label associated to the reached leaf.

The *standard* or most known model of DT is the one which uses univariate tests, usually by comparing the selected attribute with a threshold (e.g. $X_i \leq \theta$).

E. Alba et al. (Eds.): CAEPIA 2021, LNAI 12882, pp. 54–64, 2021.
https://doi.org/10.1007/978-3-030-85713-4_6

This type of DT is known as orthogonal or axis-parallel DT because of the partition of the input (attribute) space it generates. Due to the success of axis-parallel DTs there exists a plethora of machine learning algorithms to learn them from data, being C4.5 [17] and CART [5] two of the most used (both were included in the selection of the top ten algorithms in data mining [21]). However, this type of DTs cannot (directly) capture decision boundaries non parallel to the axes, having to *approximate* them by using several consecutive tests, which leads to complex (DT size) and, probably, less accurate models.

Oblique decision trees (ODT) [15] allow the use of multivariate tests in the inner nodes, obtaining in this way more compact models and, usually, better performance, at the expense of a higher time cost. Originally, a linear combination of (some of) the input variables was used as test function, $g(X_1, \ldots, X_n) \leq \theta$, but later more powerful machine learning techniques as support vector machines (SVM) or neural networks (NN) [3] have also been used.

In this work we focus on the use of SVM to obtain the test in the inner nodes, with the goal of producing an accurate single-model ODT able to manage multi-class ($k > 2$) variables. Although DTs (ODTs) are successfully used as base classifiers to build ensembles, e.g. Random Forest [6], in this first attempt we limit our study to the base ODT classifier. Our main contributions are:

- We introduce *STree*, a new SVM-based ODT. The method is able to deal with a multi-class target variable by producing a single model. The main idea behind the method is to guide the splitting process by obtaining a partition which allows to properly classifying one of the class labels with respect to (all) the remaining labels.
- An extensive experimental evaluation is carried out over a benchmark which contains 49 datasets and 5 competing algorithms. In the experiments we tested two configurations for STree, one by using a default parameterization, whose results are clearly competitive with respect to the included competing approaches, and other, with fine-tuned hyperparameters, which significantly outperforms the tested algorithms.

Our study is organized in four sections apart from this introduction. In Sect. 2 we briefly review the closer proposals to our approach, which is described in detail in Sect. 3. Section 4 contains the extensive experimental evaluation carried out over a significant benchmark, as well as its analysis. Finally in Sect. 5 we present our concluding remarks and outline future research lines.

2 Related Work

Growing a DT from data is (usually) a recursive partitioning process that splits the data into several subsets according to the test selected at each inner node. The recursive partitioning method stops when the data received by a node (mostly) belongs to the same label, then, this node becomes a leaf. Therefore, the key point in the DT learning process is how to select the test or split for an inner node. In axis-parallel DTs, information or statistical measures have been considered

to decide which test reduces most the uncertainty of the class variable. Shannon entropy (C4.5 [17]) or Gini index (CART [5]) are usual choices.

In *oblique* DTs more powerful multivariate tests are used, which leads to more compact and (usually) accurate models. However, deciding the test for a given inner node is also more computationally expensive. In most ODT algorithms the test is a linear combination of the input attributes, that is, $\beta_0 + \beta_1 X_1 + \cdots + \beta_n X_n > 0$. Then, the goal is to search for the β parameters which define the hyperplane producing the binary partition that most reduces the uncertainty of the class variable. In CART-LC [5] a coordinate descent method is used to optimize those parameters, while in OC1 [15] the method is improved by using multiple restarts and random perturbations to escape from local optima. Both, CART-LC and OC1 initialize the search with the best axis-parallel partition. In WODT [22] the optimization problem is transformed to consider the continuous and differentiable function of weighted information entropy as objective, thus, gradient descent can be used as optimization method. Furthermore, in WODT a random hyperplane is used as initialization. Metaheuristic algorithms have also been used to escape from local optima [18].

Apart from linear functions, more powerful machine learning models like neural networks or support vector machines have also been considered as split criteria [2,11], allowing in this way multivariate linear and non-linear tests. In this paper we focus on the use of support vector machines (SVM) [4,20] to build the test associated to inner nodes. The standard SVM algorithm looks for the optimal separating hyperplane which maximizes the margin of the training data regarding a binary class variable. To be able of doing this in non-linearly separable problems, we could transform the input vectors into a high dimensional feature space, where a linear classification problem is solved. These two steps can be joined in a direct computation method by using the so-called *kernel trick*, where different types of kernel can be used (linear, polynomial, etc.).

In the literature we can find several ODT approaches based on SVMs. Thus, standard SVMs with linear [2], radial-basis function [24] and polynomial [14] kernels have been used to learn the hyperplane at each inner node. More complex SVM algorithms like multisurface proximal SVM (MPSVM) are used in [13] and [23], while a twin bounded SVM (TBSVM) is used in [9]. Both, MPSVM and TBSVM learn two hyperplanes, each one being the closer to the data samples of one class and the farthest from the data samples of the other class; instances are then classified by using their distance to both hyperplanes.

The algorithms in [9,13,14,23] are able to manage multi-class problems. In [14] k, one per class, one vs rest binary problems are considered and so k SVM models learnt. Then, a vector of length k is built for each instance, where dimension i is its distance to the i-th hyperplane. Instances are then clustered in r groups by using the X-means algorithm [16]. The number of clusters, r, is determined by the X-means algorithm and is the number of branches coming out from that inner node. In [9] and [23], at each inner node, class labels are clustered into two groups by using the Bhattacharyya distance, then, these two groups are used to solve the binary classification problem by MPSVM and TBSVM respectively. Finally, in [13] at each inner node the multi-class problem

is transformed into a binary one by facing the class label with more instances against the group formed by the rest of labels.

3 Proposed Method: STree

Our goal is to design a flexible SVM Oblique Decision Tree (STree) able to cope with a multi-class target variable, $Y = \{y_1, \ldots, y_k\}$, by producing a single DT. Different SVM-ODT algorithms have been presented previously in the literature (see Sect. 2), although most of them have been directly used as base classifiers for ensemble models, without testing them as individual classifiers. The main features of our algorithm are:

- A *binary* tree is obtained, as in [9,13,23], but on the contrary to [14].
- The binary split or classification problem set in each inner node is not obtained by using clustering as in [9,14,23], but facing one class label against the rest (as in [13]). However, instead of always selecting the majority class label, we try the k one vs rest cases and choose the best one according to an impurity score, being more flexible at the expense of extra computational cost.
- As there is no agreement about which kernel is better or worse for a specific domain, we allow the use of different kernels (linear, polynomial and Gaussian radial basis function).

Our method works recursively and at each (recursive) call the algorithm receives a set of instances T. Then, the method proceeds as follows:

1. If the stopping condition (max depth, almost only one class label, etc.) is met, a leaf node is created with *outcome* equals to the more frequent label in T.
2. Otherwise, we have to split T into two groups, T^+ and T^-, in order to create the two branches for this inner node. To do this, we have to transform the multi-class problem into a binary one. Let $k' \leq k$ the number of different class labels appearing in T (notice that as the tree grows in depth, not all the labels will be present in the received set T).
   - If $k' = 2$ we already have a binary classification problem. We apply the SVM algorithm to learn the maximum margin hyperplane H and split the instances of T in T^+ and T^- accordingly to its distance to H. The hyperplane H is stored in the node.
   - If $k' > 2$ we use the well known *one-vs-the-rest* (OVR) strategy [3, pg. 339] and define k' binary classification problems: $\{y_1\}$ vs $\{y_2, \ldots, y_{k'}\}$, $\{y_2\}$ vs $\{y_1, y_3, \ldots, y_{k'}\}$, etc. Let H_i be the hyperplane learnt by the SVM algorithm for the i-th binary classification problem, T_i^+ and T_i^- be the partition of T it generates, and $impurity(Y, T)$ be a measure which evaluates the impurity of the class variable Y in T. Then, we select the hyperplane H^* such that

$$H^* = \arg \min_{i=1,..,k'} \frac{|T_i^+|}{|T|} impurity(Y, T_i^+) + \frac{|T_i^-|}{|T|} impurity(Y, T_i^-),$$

     which is stored in the node.

3. Once the hyperplane and the corresponding partition (T^+, T^-) have been selected, two branches are created for this node: positive, for those instances having positive distance to the hyperplane, and, negative, for those instances having negative distance to the hyperplane. Two recursive calls are then launched with T^+ and T^- as set of input instances respectively.

In our implementation Shannon entropy has been used as impurity measure. No pruning stage has been already designed for STree, however a pre-pruning is allowed by setting a maximum depth for the tree.

Regarding inference, for a given instance the tree must be traversed from the root to a leaf node, whose associated label is returned as outcome. At each inner node the stored hyperplane H is used and the distance of the instance to it computed. If the obtained value is ≥ 0 then the instance goes on by following the *positive* branch, otherwise it goes on by following the *negative* one.

4 Experimental Evaluation

In this Section we describe the comprehensive experiments carried out to evaluate our proposal.

4.1 Benchmark

As benchmark we have selected the same 49 datasets used in [9]: 45 of them are from the UCI machine learning repository [8] while the other 4 correspond to a problem about fecundity estimation for fisheries (see [9] for the details). The first four columns of Table 1 report the name, number of instances, attributes and class labels for each dataset.

4.2 Algorithms

The following algorithms have been included in the comparison:

– STree*. Our proposal with a fine tuned parameterization, done by using a grid search method. As mentioned in Sect. 3 our goal is to design a flexible classifier, so different hyperparameters, most of them related to the SVM learning algorithm, can be varied. In particular, in this study we have tuned the following hyperparameters:
  - Kernel. Can be linear, polynomial or Gaussian radial basis function (RBF). Default is linear.
    * Kernel coefficient gamma is optimized for polynomial and RBF kernels. Default value is $\frac{1}{n \times data.variance}$.
    * Degree is optimized for polynomial kernel. Default is 3.
  - C. Regularization hyperparameter. Default is 1.
  - Max number of iterations for the SVM (optimization) learning algorithm. Default is 10^5.

  - Max features. Number of features used to build the hyperplane. Default is all.
    In https://git.io/JYpff we show the hyperparameter values used for each dataset. When empty, the default value is selected.
- STree-default. Our proposal without fine-tuning, i.e. using the default parameterization for the SVM algorithm: kernel=linear, C = 1.0, max iterations = 10^5, and max features = all.
- TBSVM-ODT[1]. Algorithm to learn a multi-class oblique DTs by using Bhattacharyya distance-based clustering and Twin Bounded SVMs [9].
- J48SVM-ODT(See Footnote 1). Algorithm to learn a multi-class oblique DTs by using X-means clustering and SVM algorithm [14].
- WODT. Recent algorithm to learn oblique DTs based on using weighted entropy and continuous optimization [22].
- OC1. Classical algorithm to learn oblique DTs [15].
- CART. Classical algorithm to learn axis-parallel DTs [5].

In both STree* and STree-default entropy has been used as impurity measure and no maximum depth has been set, the tree grows until all the instances received in a node belong to the same class label or the hyperplane learnt by the SVM cannot separate the instances.

STree has been implemented in `python` as a `scikit-learn` classifier[2]. Publicly available versions of CART (`python/scikit-learn`) and OC1 (`C`) have been used. The code for WODT (`python`), J48SVM-ODT (`java/Weka`) and TBSVM-ODT (Matlab) have been provided by their authors. All the experiments have been run in a 3.8 GHz 8-core 10th-generation Intel Core i7 running macOS Big Sur operating system.

4.3 Results and Analysis

To evaluate the performance of each pair (algorithm, dataset), we have run a five fold cross validation 10 times (10×5cv). The same 10 seeds have been used in all the pairs to randomize the data before the cross validation. As no severe imbalance is presented in any dataset (see [9, Table A1]), accuracy is used to compare the tested algorithms performance. The mean and standard deviation over the 50 runs of the 10×5cv are reported in Table 1.

To properly analyze the results we have carried out a standard machine learning statistical analysis procedure [7,10] using the exreport tool [1]. First, a Friedman test ($\alpha = 0.05$) is performed to decide if all the algorithms are equivalent. If this hypothesis is rejected, a post hoc test is performed by using Holm's procedure ($\alpha = 0.05$) by using as control the algorithm ranked first by Friedman test.

[1] In [9] and [14] ensemble methods are proposed. In this paper we compare against the proposed base ODT classifiers.

[2] The code can be found in https://git.io/J3jkQ.

Table 1. Accuracy results (mean ± std) for all the algorithms and datasets.

#	Dataset	#S	#F	#L	STree*	STree default	WODT	J48SVM-ODT	OC1	CART	TBSVM-ODT
1	balance-scale	625	4	3	**0.9706 ± 0.015**	0.9107 ± 0.025	0.9213 ± 0.031	0.9402 ± 0.022	0.9192 ± 0.023	0.7821 ± 0.036	0.7067 ± 0.231
2	balloons	16	4	2	**0.8600 ± 0.285**	0.6533 ± 0.263	0.6783 ± 0.253	0.5950 ± 0.188	0.6200 ± 0.261	0.6833 ± 0.270	0.6050 ± 0.230
3	breast-cancer-wisc-diag	569	30	2	**0.9728 ± 0.017**	0.9689 ± 0.018	0.9659 ± 0.014	0.9518 ± 0.025	0.9335 ± 0.026	0.9239 ± 0.023	0.9657 ± 0.014
4	breast-cancer-wisc-prog	198	33	2	**0.8111 ± 0.058**	0.8021 ± 0.054	0.7076 ± 0.068	0.7240 ± 0.072	0.7100 ± 0.080	0.6915 ± 0.072	0.7449 ± 0.054
5	breast-cancer-wisc	699	9	2	0.9667 ± 0.014	0.9667 ± 0.014	0.9424 ± 0.018	**0.9672 ± 0.012**	0.9402 ± 0.021	0.9434 ± 0.021	0.9429 ± 0.017
6	breast-cancer	286	9	2	**0.7342 ± 0.048**	**0.7342 ± 0.048**	0.6632 ± 0.049	0.7074 ± 0.052	0.6497 ± 0.068	0.6363 ± 0.055	0.6564 ± 0.050
7	cardiotocography-10clases	2126	21	10	0.7915 ± 0.019	0.7915 ± 0.019	0.7793 ± 0.020	**0.8303 ± 0.017**	0.7955 ± 0.019	0.8109 ± 0.019	0.7748 ± 0.018
8	cardiotocography-3clases	2126	21	3	0.9006 ± 0.015	0.9006 ± 0.015	0.9014 ± 0.016	**0.9285 ± 0.012**	0.8998 ± 0.016	0.9205 ± 0.014	0.8967 ± 0.013
9	conn-bench-sonar-mines-rocks	208	60	2	0.7555 ± 0.068	0.7555 ± 0.068	**0.8041 ± 0.058**	0.7389 ± 0.064	0.7108 ± 0.072	0.7254 ± 0.069	0.7730 ± 0.055
10	cylinder-bands	512	35	2	0.7150 ± 0.037	0.7150 ± 0.037	0.7025 ± 0.037	**0.7265 ± 0.030**	0.6711 ± 0.042	0.7126 ± 0.046	0.6751 ± 0.036
11	dermatology	366	34	6	**0.9718 ± 0.021**	0.9661 ± 0.020	0.9642 ± 0.017	0.9571 ± 0.024	0.9161 ± 0.043	0.9396 ± 0.024	0.9707 ± 0.018
12	echocardiogram	131	10	2	**0.8148 ± 0.100**	0.8088 ± 0.070	0.7424 ± 0.080	0.8055 ± 0.084	0.7483 ± 0.087	0.7444 ± 0.083	0.7535 ± 0.072
13	fertility	100	9	2	**0.8800 ± 0.055**	0.8660 ± 0.062	0.8040 ± 0.068	0.8570 ± 0.071	0.7930 ± 0.080	0.7990 ± 0.084	0.7980 ± 0.079
14	haberman-survival	306	3	2	**0.7356 ± 0.043**	**0.7356 ± 0.043**	0.6640 ± 0.048	0.7150 ± 0.049	0.6516 ± 0.058	0.6405 ± 0.055	0.7201 ± 0.048
15	heart-hungarian	294	12	2	**0.8275 ± 0.051**	0.8177 ± 0.051	0.7619 ± 0.052	0.7850 ± 0.048	0.7583 ± 0.048	0.7506 ± 0.045	0.7798 ± 0.045
16	hepatitis	155	19	2	**0.8245 ± 0.074**	0.7961 ± 0.072	0.7684 ± 0.084	0.7619 ± 0.058	0.7568 ± 0.076	0.7658 ± 0.073	0.7737 ± 0.064
17	ilpd-indian-liver	583	9	2	**0.7235 ± 0.038**	**0.7235 ± 0.038**	0.6779 ± 0.036	0.6903 ± 0.042	0.6601 ± 0.050	0.6640 ± 0.038	0.6967 ± 0.036
18	ionosphere	351	33	2	**0.9533 ± 0.024**	0.8661 ± 0.037	0.8795 ± 0.039	0.8920 ± 0.034	0.8797 ± 0.041	0.8758 ± 0.038	0.8754 ± 0.034
19	iris	150	4	3	**0.9653 ± 0.032**	**0.9653 ± 0.032**	0.9467 ± 0.037	0.9460 ± 0.035	0.9480 ± 0.047	0.9387 ± 0.044	0.9534 ± 0.031
20	led-display	1000	7	10	0.7030 ± 0.029	0.7030 ± 0.029	0.7047 ± 0.029	**0.7209 ± 0.029**	0.6993 ± 0.031	0.7037 ± 0.030	0.7018 ± 0.024
21	libras	360	90	15	**0.7886 ± 0.052**	0.7478 ± 0.056	0.7756 ± 0.055	0.6597 ± 0.063	0.6450 ± 0.062	0.6550 ± 0.059	0.7267 ± 0.046
22	low-res-spect	531	100	9	**0.8838 ± 0.032**	0.8531 ± 0.034	0.8555 ± 0.032	0.8383 ± 0.039	0.8247 ± 0.034	0.8352 ± 0.031	0.7909 ± 0.034
23	lymphography	148	18	4	**0.8350 ± 0.059**	0.7738 ± 0.077	0.7800 ± 0.076	0.7786 ± 0.079	0.7346 ± 0.075	0.7683 ± 0.076	0.7616 ± 0.073
24	mammographic	961	5	2	0.8192 ± 0.022	0.8192 ± 0.022	0.7580 ± 0.025	**0.8215 ± 0.024**	0.7688 ± 0.042	0.7569 ± 0.021	0.7802 ± 0.026
25	molec-biol-promoter	106	57	2	0.7671 ± 0.091	0.7644 ± 0.083	**0.7789 ± 0.077**	0.7440 ± 0.091	0.7348 ± 0.079	0.7158 ± 0.088	0.6672 ± 0.089
26	musk-1	476	166	2	**0.9164 ± 0.028**	0.8435 ± 0.032	0.8412 ± 0.034	0.8269 ± 0.047	0.7764 ± 0.041	0.7769 ± 0.044	0.8340 ± 0.036
27	oocytes_merluccius_nucleus_4d	1022	41	2	**0.8351 ± 0.022**	0.8107 ± 0.025	0.7348 ± 0.026	0.7435 ± 0.034	0.7432 ± 0.039	0.7196 ± 0.031	0.7923 ± 0.024
28	oocytes_merluccius_states_2f	1022	25	3	**0.9154 ± 0.020**	**0.9154 ± 0.020**	0.9029 ± 0.022	0.9021 ± 0.022	0.8892 ± 0.022	0.8912 ± 0.025	0.9106 ± 0.018
29	oocytes_trisopterus_nucleus_2f	912	25	2	**0.8010 ± 0.022**	**0.8010 ± 0.022**	0.7491 ± 0.036	0.7540 ± 0.034	0.7477 ± 0.034	0.7258 ± 0.030	0.7619 ± 0.029
30	oocytes_trisopterus_states_5b	912	32	3	**0.9222 ± 0.018**	0.9167 ± 0.020	0.8888 ± 0.021	0.8984 ± 0.026	0.8639 ± 0.021	0.8703 ± 0.026	0.9221 ± 0.016

(*continued*)

Table 1. (*continued*)

#	Dataset	#S	#F	#L	STree*	STree default	WODT	J48SVM-ODT	OC1	CART	TBSVM-ODT
31	parkinsons	195	22	2	0.8821 ± 0.048	0.8821 ± 0.048	$\mathbf{0.9005 \pm 0.051}$	0.8446 ± 0.052	0.8656 ± 0.055	0.8559 ± 0.058	0.8792 ± 0.043
32	pima	768	8	2	$\mathbf{0.7667 \pm 0.030}$	$\mathbf{0.7667 \pm 0.030}$	0.6882 ± 0.035	0.7505 ± 0.027	0.6930 ± 0.035	0.7012 ± 0.031	0.6970 ± 0.033
33	pittsburg-bridges-MATERIAL	106	7	3	$\mathbf{0.8677 \pm 0.071}$	0.7913 ± 0.070	0.7995 ± 0.077	0.8503 ± 0.074	0.8103 ± 0.088	0.8006 ± 0.076	0.8114 ± 0.072
34	pittsburg-bridges-REL-L	103	7	3	0.6322 ± 0.101	0.6322 ± 0.101	0.6184 ± 0.085	$\mathbf{0.6450 \pm 0.104}$	0.6050 ± 0.121	0.6172 ± 0.102	0.6221 ± 0.084
35	pittsburg-bridges-SPAN	92	7	3	$\mathbf{0.6598 \pm 0.117}$	0.6302 ± 0.097	0.5872 ± 0.116	0.6149 ± 0.119	0.5793 ± 0.097	0.5575 ± 0.096	0.6302 ± 0.086
36	pittsburg-bridges-T-OR-D	102	7	2	$\mathbf{0.8616 \pm 0.069}$	$\mathbf{0.8616 \pm 0.069}$	0.8375 ± 0.074	0.8383 ± 0.078	0.8315 ± 0.083	0.8225 ± 0.089	0.8210 ± 0.072
37	planning	182	12	2	$\mathbf{0.7353 \pm 0.067}$	0.7046 ± 0.075	0.5683 ± 0.087	0.7114 ± 0.050	0.5670 ± 0.098	0.5740 ± 0.078	0.5906 ± 0.072
38	post-operative	90	8	3	$\mathbf{0.7111 \pm 0.075}$	0.6756 ± 0.090	0.5567 ± 0.106	0.7011 ± 0.087	0.5422 ± 0.128	0.5822 ± 0.106	0.5394 ± 0.103
39	seeds	210	7	3	$\mathbf{0.9529 \pm 0.028}$	0.9490 ± 0.037	0.9290 ± 0.042	0.9090 ± 0.049	0.9324 ± 0.037	0.9162 ± 0.042	0.9425 ± 0.029
40	statlog-australian-credit	690	14	2	$\mathbf{0.6783 \pm 0.039}$	0.6672 ± 0.039	0.5617 ± 0.038	0.6603 ± 0.036	0.5739 ± 0.058	0.5720 ± 0.041	$\mathbf{0.6783 \pm 0.032}$
41	statlog-german-credit	1000	24	2	$\mathbf{0.7625 \pm 0.027}$	$\mathbf{0.7625 \pm 0.027}$	0.6825 ± 0.021	0.7244 ± 0.028	0.6874 ± 0.038	0.6897 ± 0.025	0.6876 ± 0.029
42	statlog-heart	270	13	2	$\mathbf{0.8230 \pm 0.044}$	$\mathbf{0.8230 \pm 0.044}$	0.7778 ± 0.048	0.8096 ± 0.048	0.7493 ± 0.058	0.7344 ± 0.047	0.7476 ± 0.051
43	statlog-image	2310	18	7	0.9559 ± 0.010	0.9526 ± 0.008	0.9534 ± 0.011	$\mathbf{0.9676 \pm 0.011}$	0.9501 ± 0.009	0.9613 ± 0.009	0.9536 ± 0.043
44	statlog-vehicle	846	18	4	$\mathbf{0.7930 \pm 0.030}$	$\mathbf{0.7930 \pm 0.030}$	0.7297 ± 0.030	0.7323 ± 0.031	0.7085 ± 0.037	0.7064 ± 0.031	0.7896 ± 0.029
45	synthetic-control	600	60	6	0.9500 ± 0.025	0.9388 ± 0.030	$\mathbf{0.9785 \pm 0.015}$	0.9223 ± 0.031	0.8632 ± 0.037	0.9023 ± 0.025	0.9716 ± 0.015
46	tic-tac-toe	958	9	2	$\mathbf{0.9844 \pm 0.008}$	0.9833 ± 0.008	0.9423 ± 0.033	0.9833 ± 0.008	0.9185 ± 0.037	0.9523 ± 0.019	0.9749 ± 0.012
47	vertebral-column-2clases	310	6	2	$\mathbf{0.8529 \pm 0.041}$	$\mathbf{0.8529 \pm 0.041}$	0.8071 ± 0.056	0.8523 ± 0.045	0.8152 ± 0.047	0.7997 ± 0.042	0.8226 ± 0.040
48	wine	178	13	3	$\mathbf{0.9792 \pm 0.022}$	0.9758 ± 0.024	0.9679 ± 0.028	0.9786 ± 0.021	0.9162 ± 0.047	0.9016 ± 0.053	0.9775 ± 0.021
49	zoo	101	16	7	$\mathbf{0.9575 \pm 0.045}$	0.9476 ± 0.042	0.9514 ± 0.044	0.9238 ± 0.047	0.8910 ± 0.077	0.9556 ± 0.044	0.9363 ± 0.049

We carried out two statistical analysis, including or not algorithm STree*. When all the algorithms are included, Friedman test reports a p-value of 4.9919e−27, thus rejecting the null hypothesis that all the algorithms are equivalent. The results of the post hoc Holm's tests are shown in Table 2(a) by using STree* as control. The columns *rank* and p-value represent the ranking obtained by the Friedman test and the p-value adjusted by Holm's procedure, respectively. The columns *win*, *tie* and *loss* contain the number of times that the control algorithm wins, ties and loses with respect to the row-wise algorithm. The p-values for the non-rejected null hypothesis are boldfaced. As can be observed (finely tuned) STree* algorithm significantly outperforms the remaining algorithms. In order to compare our proposal without tuning the hyperparameters we have repeated the statistical analysis excluding STree*. Again Friedman test rejects the hypothesis of all the algorithms being equivalent (p-value 2.8244e−16). The post hoc analysis is shown in Table 2(b), where STree-default is taken as control, as it is now the algorithm ranked first by Friedman test. As we can observe, when the hyperparameters are not tuned for STree, STree-default significantly outperforms all the algorithms but J48SVM-ODT, there being no statistically significant difference in this case.

Table 2. Results of the post-hoc test for the mean accuracy of the algorithms.

Method	rank	p-value	win	tie	loss
STree*	1.65	–	–	–	–
STree_default	2.83	7.172e−03	28	21	0
J48SVM-ODT	3.49	5.142e−05	41	0	8
TBSVM-ODT	4.34	2.338e−09	46	1	2
WODT	4.37	1.999e−09	43	0	6
CART	5.45	1.696e−17	45	0	4
OC1	5.88	2.211e−21	48	0	1

(a) With STree*

Method	rank	p-value	win	tie	loss
STree_default	2.04	–	–	–	–
J48SVM-ODT	2.65	**1.053e−01**	34	0	15
TBSVM-ODT	3.39	7.314e−04	40	0	9
WODT	3.49	3.788e−04	34	0	15
CART	4.53	1.791e−10	41	0	8
OC1	4.90	2.026e−13	45	0	4

(b) Without STree*

Finally, we have also analyzed the complexity (size) of the obtained trees (see https://git.io/J3jI6) and also the training time of the different algorithms (see https://git.io/J3jI9). Table 3 shows the results on average over the 49 datasets once we normalize them by using STree-default as control. Regarding size we can observe that STree obtains the smallest trees, while J48SVM-ODT and WODT, which are the closer algorithms to STree, obtain trees with twice the nodes of STree-default. Regarding time, although this is not a fair comparison because of the different implementations, we can observe that apart from axis-parallel CART, STree algorithms are strongly competitive, with only TBSVM-ODT being faster than STree-default, but that algorithm is so far in accuracy.

Table 3. Datasets used during the experimentation

	STree*	STree-default	WODT	J48SVM-ODT	OC1	CART	TBSVM-ODT
Size	0.87	1.00	19.18	2.17	2.00	8.58	3.52
CPU time	0.85	1.00	7.96	3.61	7.13	0.09	0.88

5 Conclusion

A new algorithm to build oblique DT able to directly manage a multi-class target variable has been presented. The algorithm produces a binary DT that needs to learn several hyperplanes at each split, although only one is stored for inference. The experiments show that the proposal works well over a great range of domains (49 datasets) and its performance is remarkable when compared against 5 competing algorithms. We also observe that tuning the hyperparameters of the SVM algorithm for each dataset is key to obtain better results, leading to an STree version that significantly outperforms all the competing tested methods.

As future research we plan to work with sub-spaces (few variables) instead of considering all the features at each node of the ODT. These variables can be selected by using some univariate or multivariate filter feature selection algorithm, or randomly following the random-subspace principle [6]. Aggregating this weaker classifiers by using ensemble-based techniques is also of interest, as has been done in [9,14]. Finally, the advantage of a fine tuning of the SVM hyperparameters has revealed to be key in the performance of the proposed algorithm, at the expense of a high computational CPU time requirement. As future research we plan to study some type of light *auto-tuning* that can be carried out embedded in the STree algorithm.

Acknowledgements. We are indebted to the authors of [9,14] and [22] because of providing us with the code of their implementations. This work has been partially funded by FEDER funds, the JCCM Government and the Spanish Goverment through the projects SBPLY/17/180501/ 000493 and PID2019-106758GB-C33.

References

1. Arias, J., Cózar, J.: ExReport: fast, reliable and elegant reproducible research (2016). https://cran.r-project.org/web/packages/exreport/index.html. Accessed 04 Aug 2021
2. Bennett, K.P., Blue, J.A.: A support vector machine approach to decision trees. In: 1998 IEEE International Joint Conference on Neural Networks Proceedings, vol. 3, pp. 2396–2401 (1998)
3. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer, New York (2006)
4. Boser, B.E., Guyon, I.M., Vapnik, V.N.: A training algorithm for optimal margin classifiers. In: Proceedings of the Fifth Annual Workshop on Computational Learning Theory, COLT 1992, pp. 144–152 (1992)

5. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and Regression Trees. Wadsworth and Brooks, Monterey (1984)
6. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001). https://doi.org/10.1023/A:1010933404324
7. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. **7**, 1–30 (2006)
8. Dua, D., Graff, C.: UCI machine learning repository (2017). http://archive.ics.uci.edu/ml
9. Ganaie, M., Tanveer, M., Suganthan, P.: Oblique decision tree ensemble via twin bounded SVM. Expert Syst. Appl. **143**, 113072 (2020)
10. García, S., Herrera, F.: An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. J. Mach. Learn. Res. **9**, 2677–2694 (2008)
11. Kontschieder, P., Fiterau, M., Criminisi, A., Bulò, S.R.: Deep neural decision forests (2015)
12. Kotsiantis, S.B.: Decision trees: a recent overview. Artif. Intell. Rev. **39**(4), 261–283 (2013). https://doi.org/10.1007/s10462-011-9272-4
13. Manwani, N., Sastry, P.S.: Geometric decision tree. IEEE Trans. Syst. Man Cybern. Part B **42**(1), 181–192 (2012)
14. Menkovski, V., Christou, I.T., Efremidis, S.: Oblique decision trees using embedded support vector machines in classifier ensembles (2008)
15. Murthy, S.K., Kasif, S., Salzberg, S.: A system for induction of oblique decision trees. J. Artif. Intell. Res. **2**(1), 1–32 (1994)
16. Pelleg, D., Moore, A.W.: X-means: extending k-means with efficient estimation of the number of clusters. In: Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), pp. 727–734 (2000)
17. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers Inc., San Francisco (1993)
18. Rivera-Lopez, R., Canul-Reich, J., Gámez, J.A., Puerta, J.M.: OC1-DE: a differential evolution based approach for inducing oblique decision trees. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2017. LNCS (LNAI), vol. 10245, pp. 427–438. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59063-9_38
19. Shotton, J., et al.: Efficient human pose estimation from single depth images. IEEE Trans. Pattern Anal. Mach. Intell. **35**(12), 2821–2840 (2013)
20. Vapnik, V.N.: The Nature of Statistical Learning Theory. Springer, Heidelberg (1995). https://doi.org/10.1007/978-1-4757-2440-0
21. Wu, X., et al.: Top 10 algorithms in data mining. Knowl. Inf. Syst. **14**(1), 1–37 (2007). https://doi.org/10.1007/s10115-007-0114-2
22. Yang, B.B., Shen, S.Q., Gao, W.: Weighted oblique decision trees. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 5621–5627 (2019)
23. Zhang, L., Suganthan, P.N.: Oblique decision tree ensemble via multisurface proximal support vector machine. IEEE Trans. Cybern. **45**(10), 2165–2176 (2015)
24. Zhang, L., Zhou, W., Su, T., Jiao, L.: Decision tree support vector machine. Int. J. Artif. Intell. Tools **16**(1), 1–16 (2007)