Classifying very-high-dimensional data with random forests of oblique decision trees

Thanh-Nghi Do, Philippe Lenca, Stéphane Lallich, and Nguyen-Khang Pham

Abstract The random forests method is one of the most successful ensemble methods. However, random forests do not have high performance when dealing with very-high-dimensional data in presence of dependencies. In this case one can expect that there exist many combinations between the variables and unfortunately the usual random forests method does not effectively exploit this situation. We here investigate a new approach for supervised classification with a huge number of numerical attributes. We propose a random oblique decision trees method. It consists of randomly choosing a subset of predictive attributes and it uses SVM as a split function of these attributes. We compare, on 25 datasets, the effectiveness with classical measures (e.g. precision, recall, F1-measure and accuracy) of random forests of random oblique decision trees with SVMs and random forests of C4.5. Our proposal has significant better performance on very-high-dimensional datasets with slightly better results on lower dimensional datasets.

Thanh-Nghi Do
Institut Telecom; Telecom Bretagne
UMR CNRS 3192 Lab-STICC
Université européenne de Bretagne, France
Can Tho University, Vietnam
e-mail: tn.do@telecom-bretagne.eu

Philippe Lenca
Institut Telecom; Telecom Bretagne
UMR CNRS 3192 Lab-STICC
Université européenne de Bretagne, France
e-mail: philippe.lenca@telecom-bretagne.eu

Stéphane Lallich
Université de Lyon, Laboratoire ERIC, Lyon 2, France
e-mail: stephane.lallich@univ-lyon2.fr

Nguyen-Khang Pham
IRISA, Rennes, France
Can Tho University, Vietnam
e-mail: pnguyenk@irisa.fr

1 Introduction

Since the nineties the machine learning community studies how to combine multiple classifiers into an ensemble of classifiers to build models that are more accurate than a single one. The purpose of ensemble classifiers is to reduce the variance and/or the bias in learning algorithms. Bias is the systematic error term (independent of the learning sample) and variance is the error due to the variability of the model with respect to the learning sample randomness. Buntine [Buntine, 1992] introduced Bayesian techniques for tree averaging to reduce the variance in learning methods. Stacking method [Wolpert, 1992] aims at minimizing the bias of learning algorithms. Freund and Schapire proposed Boosting [Freund and Schapire, 1995] to simultaneously reduce the bias and the variance while the Bagging method proposed by Breiman [Breiman, 1996] reduces the variance of a learning algorithm without increasing its bias too much.

The random forests approach proposed by Breiman [Breiman, 2001] has been one of the most successful ensemble methods. Random forests algorithm creates a collection of unpruned decision trees (built so that at each node the best split is done from a randomly chosen subset of attributes) from bootstrap samples (sampling with replacement from the original dataset). The generalization error of a forest depends on the strength of the individual trees in the forest and on the dependence between them. Random forest algorithm constructs unpruned trees for keeping low bias and uses the randomization for controlling high diversity between trees in the forest. Two classifiers are diverse if they make different errors on new data points [Dietterich, 2000a]. Random forests approach gives high accuracy compared with state-of-the-art supervised classification algorithms, including AdaBoost [Freund and Schapire, 1995] and SVM [Vapnik, 1995]. As mentioned in [Breiman, 2001] random forests method is fast, robust to noise and does not overfit unlike AdaBoost algorithm which is sensitive to noisy data [Dietterich, 2000b]. Random forests algorithm has been shown to build accurate models with practical relevance for classification, regression and novelty detection [Breiman, 2001].

The tree construction of habitual random forests only picks, at each node, a single attribute for node splitting. Thus the individual trees are less efficient when dealing with data having dependencies among attributes, as it could be the case with very-high-dimensional datasets.

In this paper which is an extended version of [Do et al., 2009], we propose to use linear proximal SVMs [Fung and Mangasarian, 2001] for performing multivariate node splitting during the tree construction (in order to use dependencies between attributes), producing individual classifiers that are stronger than in the usual forests. Numerical test results on UCI [Asuncion and Newman, 2007], Statlog [Michie et al., 1994] and very-high-dimensional Bio-medical [Jinyan and Huiqing, 2002] datasets show that our random forests of oblique decision trees are often more accurate than random forests of C4.5 [Quinlan, 1993] and SVM in terms of precision, recall, F1-measure and accuracy [van Rijsbergen, 1979]. In particular our proposal has significant better

performance on very-high-dimensional data with better -but not significant- results on lower dimensional datasets.

The paper is organized as follows. Section 2 briefly introduces random forests and our random forests of oblique decision trees for classification. The experimental results are presented in Section 3. We then conclude in Section 4.

2 Random forests of oblique decision trees

In early bagging approach proposal of Breiman [Breiman, 1996], an ensemble of decision trees is built from bootstrap samples drawn with replacement from the original dataset. Then, the predictions of these trees are aggregated, by a majority vote in classification tasks or by an average for regression problems. Ho [Ho, 1995] also proposed the random subspace method which randomly selects a subset of attributes for growing each tree. Amit and Geman [Amit and Geman, 2001] used a random selection of attributes for the search of the best split at each node. Finally, these approaches were extended and formalized in the term of random forests by Breiman [Breiman, 2001].

2.1 Random forests

The random forests algorithm of Breiman in [Breiman, 2001] aims at creating a collection of high performance decision trees with high diversity between individual trees in the forest. He proposed to use two strategies to keep low bias and low dependence between trees in the forest. For reaching out to low bias, he proposed to build the individual trees without pruning, i.e. which are grown to maximum depth. To the diversity control of the trees, he also proposed to uses a bootstrap replica from the original training set to construct the trees and randomly choose a subset of attributes on which to base the calculation of the best split at a decision node.

Let us consider the classification task with m datapoints $x_i (i = 1, m)$ and n attributes, a decision tree (denoted by DT) in a random forest of k trees (denoted by RF$= \{DT_i\}_{i=1,k}$) is constructed as follows:

- The training set is a bootstrap replica of m individuals, i.e. a random sampling with replacement from the original training set.
- For each node of the tree, randomly choose n' attributes ($n' << n$, e.g. $n' = \sqrt{n}$) and calculate the best split based on one of these n' attributes.
- The tree is grown to its maximal depth without pruning.

To classify a new individual, the prediction phase uses an unweighted majority vote of the trees for a classification task or an average-up for a regression task. Breiman proposed to use datapoints of out-of-bag (about 36.8% of the original training set are out of the bootstrap sample) to estimate important attributes and the error

in the forest while adding a new tree. The random forests algorithm exhibits high accuracy. Furthermore, it is also fast, robust to noise and does not overfit. Breiman also extended random forests for unsupervised learning tasks [Breiman, 2001].

Recently, Robnik-Sikonja proposed in [Robnik-Sikonja, 2004] some possibilities for improving random forests. He investigated strategies to increase strength or to increase diversity of individual trees in the forest. He used several attribute evaluation measures instead of just one. He also proposed the use of weighted voting.

An another idea proposed by Geurts et al. [Geurts et al., 2006] aims at building totally random trees. They proposed and studied an algorithm called Extra-Trees. It uses the whole training set instead of a bootstrap replica to build the trees. At each level of the tree, the method randomly chooses an attribute to split the data (if it is an continuous attribute then the cut-point is also chosen randomly), i.e. independently of the class labels. As mentioned in [Geurts et al., 2006] the explicit randomization of the splitting in the Extra-Trees could reduce variance more strongly than the weaker randomization schemes used by habitual random forests. Obviously, the algorithm is very fast for training, but the strength of the individual trees in the forest may be reduced. The algorithm Extra-Trees is close to the algorithm PERT (for perfect random tree ensembles) proposed in [Cutler and Guohua, 2001].

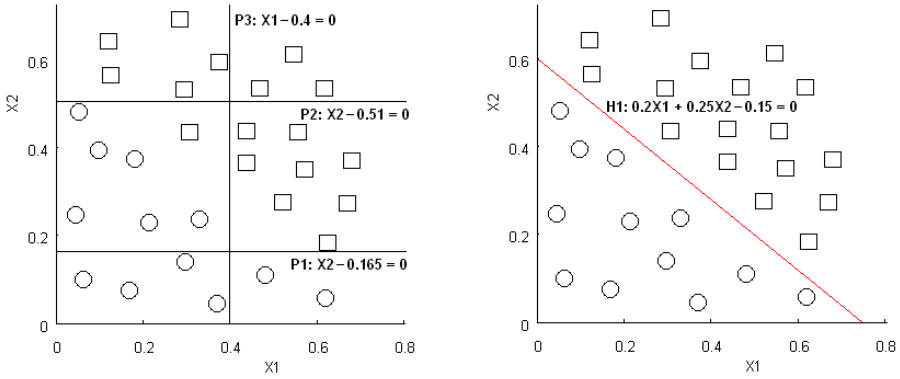

Fig. 1 Single attribute (left) and bi-variate (right) node splitting

2.2 Oblique decision trees

The tree construction of habitual random forests picks a single attribute for node splitting [Breiman et al., 1984], [Quinlan, 1993], [Ho, 1995], [Amit and Geman, 2001], [Cutler and Guohua, 2001], [Robnik-Sikonja, 2004], [Geurts et al., 2006]. Thus, the strength of individual trees is reduced, particularly when dealing with datasets having dependencies among attributes. For example in figure 1, any univariate splitting (perpendicular to axes) can not totally separate

the data into two classes but the bi-variate splitting, i.e. H_1 (combination of two attributes) perfectly classifies the data into two classes (some ensemble methods can deal with this problem if they use a large number of trees, see for example [Cutler and Guohua, 2001] and [Geurts et al., 2006]). Therefore, the univariate splitting used by the usual tree construction is not robust in this case.

At the opposite, multivariate splitting criteria -where several attributes may participate in a single node split test-, may dramatically improve the trees performance. The problem of constructing an oblique decision tree is well known to be NP-hard [Heath, 1992]. Most of the multivariate splitting criteria are based on linear combination of the input attributes. As pointed out by [Rokach and Maimon, 2005] finding the best linear combination can be achieved in different ways. For example, linear programming [Bennett and Mangasarian, 1994], linear discriminant analysis [Loh and Vanichsetakul, 1988, Yildiz and Alpaydin, 2005] or linear combinations of attributes [Breiman et al., 1984]. With multivariate splitting criteria each test is equivalent to a hyperplane with an oblique orientation to the axes. Because of the computational intractability of finding an optimal orientation for these hyperplanes, heuristic methods were proposed to produce good trees like in the algorithm OC1 [Murthy et al., 1993, Murthy et al., 1994]. Indeed, the greedy approaches can deal only with low dimensional datasets due to combinatorial explosion.

The OC1 approach was extended by Wu et al. [Wu et al., 1999] by modifying the splitting criterion of the basic OC1 algorithm or by post-processing OC1 output. While these modifications outperform the basic OC1 on the correctness and the robustness to noise, the optimal hyperplanes are found with standard SVMs through the resolution of a quadratic programming. Therefore, the proposed approach has a high cost for the learning task.

Our investigation aims at performing multivariate node splitting during tree construction, thus producing individual oblique classifiers that are stronger than the usual random forests. As a whole our method combines both advantages of oblique splitting and ensemble methods in an efficient manner.

2.3 Random forests of oblique decision trees

Our random forest algorithm constructs a collection of oblique decision trees (denoted by RF-ODT) in the same framework of random forests proposed by Breiman [Breiman, 2001]. The main difference is that each random oblique decision tree (ODT) in the forest (RF-ODT$=\{ODT_i\}_{i=1,k}$) uses linear SVMs for performing multivariate node splitting as proposed in [Do et al., 2009]. Our proposal is thus an hybridization of decision trees with SVMs. SVMs are here used in the growing phase to create the oblique trees.

Others works proposed hybridization in a post-growing phase, to attach classifiers to the tree's leaves as for example, genetic algorithm [Carvalho and Freitas, 2004], neural network [Zhou and Chen, 2002, Maji, 2008], SVM [Xu et al., 2006] or multiple-classifier [Cohen et al., 2007]. Recently

[Simon et al., 2009] proposed to embedded multiple proximal SVM into a binary tree architecture for multi-classes problem.

We propose to use linear proximal SVMs [Fung and Mangasarian, 2001] to build oblique splits on randomly chosen attributes because they are very fast for training and give good accuracy when compared with standard SVMs (see for example the experiments by [Do and Poulet, 2006] where one million datapoints in 20-dimensional input space are classified into two classes in 13 seconds on a PC (2.4 GHz Pentium IV, 512 MB RAM)).

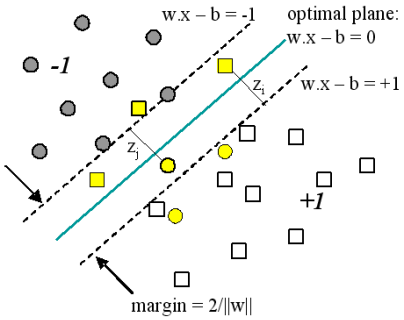

Fig. 2 Linear separation of the datapoints into two classes

Briefly, consider the linear binary classification task depicted in figure 2, with m datapoints $x_i (i = 1, m)$ in n dimensions (attributes). It is represented by the $[m \times n]$ matrix A, having corresponding labels $y_i = \pm 1$, denoted by the $[m \times m]$ diagonal matrix D of ± 1 (where $D[i,i] = 1$ if x_i is in class +1 and $D[i,i] = -1$ if x_i is in class -1). A SVM algorithm tries to find the best plane to separate the classes, i.e. the one farthest from both class +1 and class -1. Any point falling on the wrong side of its supporting plane is considered to be an error. Therefore, SVMs simultaneously maximize the distance between two parallel supporting planes for each class and minimize the errors.

Classical SVMs pursue these goals with the quadratic program (1):

$$\min_{w,\ b,\ z} \psi(w,b,z) = (1/2)\|w\|^2 + cz \qquad (1)$$
$$s.t. : D(Aw - eb) + z \geq e$$

where e is the column vector of 1, $z \in R^m$ is the non negative slack vector, and $c \in R^1$ a positive constant; w and b be the normal vector and the scalar of the plane respectively; z and c are used to tune errors and margin size.

The plane (w,b) is obtained by solving the quadratic programming (1). Then, the classification function of a new datapoint x based on the plane is:

$predict(x) = sign(w.x - b)$

Unfortunately, the computational cost requirements of the SVM solutions in (1) are at least the square of the number of training datapoints, making classical SVM intractable for large datasets. The proximal SVM proposed by Fung and Mangasarian [Fung and Mangasarian, 2001] modified the quadratic programming (1) by using the equality instead of the inequality constraints and a least squares 2-norm error in the objective function ψ. They also changed the formulation of the margin maximization to the minimization of $1/2\|w,b\|^2$. Thus substituting for z from the constraint in terms of w and b into the objective function ψ of the quadratic programming (1) yields an unconstrained problem (2):

$$\min_{w,\ b} \Psi(w,b) = (1/2)\|w,b\|^2 + (c/2)\|e - D(Aw - eb)\|^2 \tag{2}$$

In the optimal configuration for (2), the gradient with respect to w and b should be zero. This yields the linear equation system of $(n+1)$ variables $(w_1,\ w_2,\ \ldots,\ w_n,\ b)$ as follows:

$$(w_1,\ w_2,\ \ldots,\ w_n,\ b)^T = (\frac{1}{c}I + E^T E)^{-1} E^T De \tag{3}$$

where $E = [A \quad -e]$, I denotes the identity matrix.

The proximal SVM formulation (3) requires thus only the solution of linear equations of $(n+1)$ variables $(w_1, w_2,\ \ldots, w_n, b)$ instead of the quadratic programming (1). Its complexity is linear with the number of training datapoints. If the dimension of input space is small enough (less than 10^4), even if there are millions of datapoints, the proximal SVM algorithm is able to classify them in an efficient manner. Numerical test results have shown that this algorithm gives similar accuracy compared to standard SVM like LibSVM [Chang and Lin, 2001] but the proximal SVMs are much faster than standard SVMs. Another non-standard SVM, the Least-Squares SVM proposed by [Suykens and Vandewalle, 1999] also replaces standard SVM optimization inequality constraints with equalities; so its performance is very close to the proximal SVM. Our proposal is thus efficient in comparison with methods discussed in this paper.

Therefore, we propose to use proximal SVMs for performing multivariate node splitting during oblique trees construction. Furthermore, at a decision node of the tree, the n' randomly chosen attributes ($n' << n$, e.g. $n' = \sqrt{n}$) brings out to low-dimensional problems where the proximal SVM algorithm is very fast compared with other ones. In addition, the costs re-balancing method [Veropoulos et al., 1999] is also used to deal with the class imbalance problem at multivariate node splitting. The random forests of oblique trees algorithm builds an ensemble of unpruned oblique trees according to the classical top-down procedure (see Figure 3). Note that our algorithm not only improves the strength of the individual trees in the forest using oblique splitting during tree construction but also keeps the high diversity between them as it is done with usual random forests method. It means that our forests

create a collection of random oblique trees using a bootstrap replica from the original training set to construct oblique trees and a random subset of attributes on which to build multivariate splitting at each node.

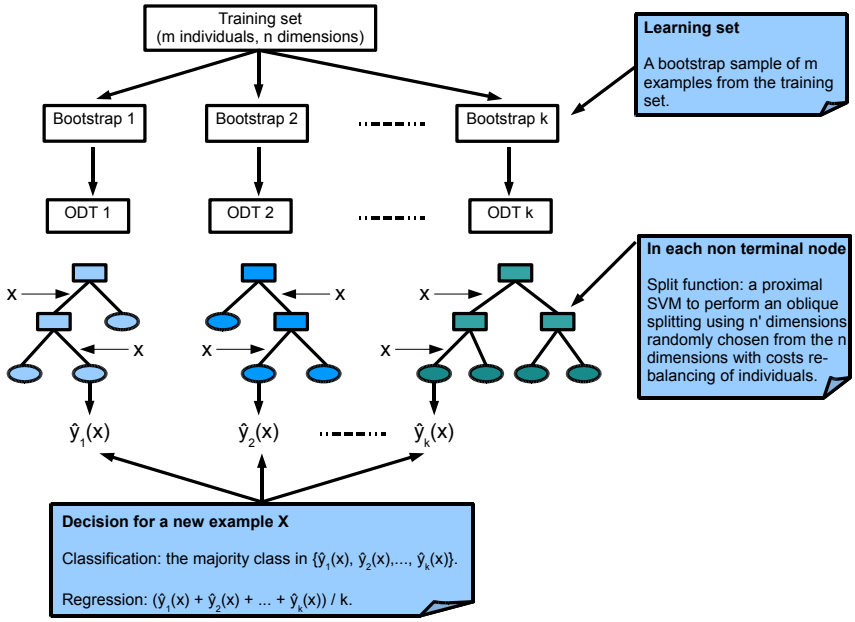

Fig. 3 Random Forest of Oblique Decision Trees

3 Evaluation

We here report the comparisons of the performance of random forests of oblique trees, of random forests of C4.5 and of SVMs. Random forests of oblique decision trees (RF-ODT) and random forests of C4.5 decision trees (using C4.5 program by [Quinlan, 1993]) have been implemented in C++ and C respectively. We also use the highly efficient standard SVM algorithm LibSVM [Chang and Lin, 2001].

In order to test the statistical signification of the observed results we used both the Student test and the sign test. Indeed, these two tests give complementary point of views on the data.

3.1 Experiments setup

The experimental setup used fifteen very-high-dimensional datasets from the Biomedical repository [Jinyan and Huiqing, 2002] and ten standard datasets from UCI [Asuncion and Newman, 2007] and Statlog [Michie et al., 1994] repositories.

In order to evaluate performance for binary classification tasks, we pre-processed multi-class (more than two classes, denoted by an asterisk in the tables 1 and 2) datasets as two-class problems. In the tables 1 and 2 the fourth column shows how we convert multi-class to two-class (for example with the OpticDigits dataset, the digit "0" is mapped to the +1 class and the remaining digits are considered as the -1 class). The performance of the classification algorithms is analyzed in terms of precision, recall, F1-measure and accuracy. The test protocols are presented in the last column of the tables 1 and 2. With datasets having training set (trn) and testing set (tst) available, we used the training data to tune the parameters of the algorithms for obtaining a good accuracy in the learning phase. For Random forests, we tuned the number of trees in the forests and the number of random attributes at each node splitting. For LibSVM, we tuned the positive constant c for tradeoff of errors and the margin size ($c = 10^5$ for the 15 high dimensional datasets). Then the obtained model is evaluated on the test set. If the training set and testing set are not available then we used cross-validation protocols to evaluate the performance. With datasets having less than three hundred datapoints, the test protocol is leave-one-out cross-validation (loo). It involves using a single datapoint from the dataset as the testing data and the remaining datapoints as the training data. This is repeated such that each datapoint in the dataset is used once as the testing data. With dataset having more than three hundred datapoints, 10-fold cross-validation is used to evaluate the performance. The dataset is partitioned into 10 folds. A single fold is retained as the validation set, and the remaining 9 folds are used as training data. The cross-validation process is then repeated 10 times (folds). The results from the 10 folds are then averaged to produce the final result.

Table 1 Description of very-high-dimensional datasets

ID	Datasets	#Datapoints	#Dimensions	Classes	Protocols
1	Colon Tumor	62	2000	tumor, normal	loo
2	ALL-AML-Leukemia	72	7129	ALL, AML	trn-tst
3	*MLL-Leukemia	72	12582	MLL, rest	trn-tst
4	Breast Cancer	97	24481	relapse, non-relapse	trn-tst
5	Duke Breast Cancer	42	7129	cancer, normal	loo
6	Prostate Cancer	136	12600	cancer, normal	trn-tst
7	Lung Cancer	181	12533	cancer, normal	trn-tst
8	Central Nervous System	60	7129	positive, negative	loo
9	Translation Initiation Site	13375	927	positive, negative	10-fold
10	Ovarian Cancer	253	15154	cancer, normal	loo
11	Diffuse Large B-Cell Lymphoma	47	4026	germinal, activated	loo
12	*Subtypes of Acute Lymphoblastic (Hyperdip)	327	12558	Hyperdip, rest	trn-tst
13	*Subtypes of Acute Lymphoblastic (TEL-AML1)	327	12558	TEL-AML1, rest	trn-tst
14	*Subtypes of Acute Lymphoblastic (T-ALL)	327	12558	TEL-ALL, rest	trn-tst
15	*Subtypes of Acute Lymphoblastic (Others)	327	12558	Others, diagnostic groups	trn-tst

Table 2 Description of standard datasets

ID	Datasets	#Datapoints	#Dimensions	Classes	Protocols
16	Bupa	345	6	1, 2	10-fold
17	Breast Cancer Wisconsin	569	30	M, B	10-fold
18	Pima	768	8	1, 2	10-fold
19	*Segment	2310	19	1, rest	10-fold
20	Spambase	4601	57	spam, non-spam	10-fold
21	*Opticdigits	5620	64	0, rest	trn-tst
22	*Satimage	6435	36	1, rest	trn-tst
23	*Pendigits	10992	16	9, rest	trn-tst
24	*Letters	20000	16	A, rest	10-fold
25	*Shuttle	58000	9	1, rest	trn-tst

3.2 Classification results on very-high-dimensional datasets

For dealing with very-high-dimensional datasets we varied the size of the forest (k) from 50 to 500 trees and the number of random attributes (n') for each node of the tree from 100 to 500. Then the good parameter values were chosen (table 3). With the standard SVM algorithm LibSVM, the linear kernels are appropriate for very-high-dimensional datasets having a very large number of dimensions and few datapoints.

The main result of the carried out experiments (table 4) is that RF-ODT outperforms RF-C4.5 and LibSVM. Overall, using paired Student ratio test, one can see that RF-ODT significantly improves the mean accuracy (table 5) of 3.6 percent points compared to RF-C4.5 (p-value $= 0.0462$) and 6.4 points compared to LibSVM (p-value $= 0.0204$). The comparison dataset by dataset using the sign test shows that on the 15 datasets, RF-ODT systematically prevails on RF-C4.5 (9 wins, 6 ties, 0 defeat, p-value $= 0.0039$) and is beaten once only by LibSVM (10 wins, 4 ties, 1 defeat, p-value $= 0.0117$).

For a more detailed assessment of the performance of RF-ODT facing RF-C4.5 and LibSVM, in addition to the error rate, we also calculated the precision, the recall and the F1-measure [van Rijsbergen, 1979]. The precision for a class is the number of datapoints correctly labeled as belonging to the class divided by the total number of datapoints labeled as belonging to the class. The recall for a class is the number of datapoints correctly labeled as belonging to the class divided by the total number of elements that actually belong to the class. The F1-measure is a synthesis of the precision and the recall, which is defined as the harmonic mean of these both quantities. Compared to the arithmetic mean, the harmonic mean has the particularity to be more sensitive to the minimum of the precision and the recall.

Regarding the comparison of RF-ODT with RF-C4.5, one can see that the gain ensured by RF-ODT is above all due to the increase of the recall (table 7) which is significantly improved of 7.1 percent points on average (Student p-value $= 0.0296$). The comparison of the recalls, dataset by dataset, shows that RF-ODT is beaten only once by RF-C4.5 (8 wins, 6 ties, 1 defeat, p-value $= 0.0391$). The empirical gain on the precision (table 6), which worth 3 percent points, due to the excellent performance of RF-ODT on the datasets 6 and 8, is not significant. The dataset by dataset

Table 3 Parameter values of random forest algorithms

ID	Datasets	#Random dimensions	#Trees
1	Colon Tumor	100	200
2	ALL-AML-Leukemia	100	300
3	*MLL-Leukemia	500	100
4	Breast Cancer	100	500
5	Duke Breast Cancer	200	500
6	Prostate Cancer	250	100
7	Lung Cancer	250	100
8	Central Nervous System	500	100
9	Translation Initiation Site	150	200
10	Ovarian Cancer	500	100
11	Diffuse Large B-Cell Lymphoma	150	200
12	*Subtypes of Acute Lymphoblastic (Hyperdip)	150	500
13	*Subtypes of Acute Lymphoblastic (TEL-AML1)	150	500
14	*Subtypes of Acute Lymphoblastic (T-ALL)	200	100
15	*Subtypes of Acute Lymphoblastic (Others)	500	500
16	Bupa	4	50
17	Breast Cancer Wisconsin	10	50
18	Pima	5	50
19	*Segment	10	50
20	Spambase	20	50
21	*Opticdigits	20	50
22	*Satimage	6	200
23	*Pendigits	8	50
24	*Letters	8	50
25	*Shuttle	5	50

Table 4 Classification results on very-high-dimensional datasets

Dataset	Precision			Recall			F1-measure			Accuracy		
ID	LibSVM	RF-C4.5	RF-ODT	LibSVM	RF-C4.5	RF-ODT	LibSVM	RF-C4.5	RF-ODT	LibSVM	RF-C4.5	RF-ODT
1	68.18	76.19	**82.61**	75.00	72.73	**86.36**	71.43	74.42	**84.44**	80.65	82.26	**88.71**
2	**100**	95.24	95.24	95.00	**100**	**100**	97.44	**97.56**	**97.56**	**97.06**	**97.06**	**97.06**
3	75.00	**100**	**100**	**100**	**100**	**100**	85.71	**100**	**100**	93.33	**100**	**100**
4	69.23	83.33	**84.62**	75.00	83.33	**91.67**	72.00	83.33	**88.00**	63.16	78.94	**84.21**
5	85.00	**94.12**	90.00	**94.44**	80.00	90.00	89.47	86.49	**90.00**	**90.48**	88.10	**90.48**
6	73.53	75.76	**100**	**100**	**100**	96.00	84.75	86.21	**97.96**	73.53	76.47	**97.06**
7	88.26	**93.75**	**93.75**	**100**	**100**	**100**	93.75	**96.77**	**96.77**	98.66	**99.33**	**99.33**
8	47.62	45.46	**61.91**	55.56	23.81	**61.91**	51.28	31.25	**61.91**	68.33	63.33	**73.33**
9	83.13	**92.58**	90.78	**84.42**	73.83	79.75	83.77	82.15	**84.91**	92.15	92.30	**93.20**
10	**100**	98.78	**100**	**100**	**100**	**100**	**100**	99.39	**100**	**100**	99.21	**100**
11	91.30	**95.65**	92.00	87.50	91.67	**95.83**	89.36	93.62	**93.88**	89.36	**93.62**	**93.62**
12	95.46	95.24	**100**	**95.46**	90.91	**95.46**	95.46	93.02	**97.67**	98.21	97.32	**99.11**
13	**100**	**100**	**100**	**100**	96.30	96.30	**100**	98.11	98.11	**100**	99.11	99.11
14	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
15	92.59	**100**	**100**	39.68	29.63	**55.56**	55.56	45.71	**71.43**	64.29	83.93	**89.29**

results (6 wins, 6 ties, 3 defeat, p-value = 0.5078) support those comments. As results in table 8, the F1-measure obtained from RF-ODT is significantly improved by 6.3 points on average compared with the F1-measure obtained from RF-C4.5 (p-value = 0.0269).

Table 5 Accuracy comparison on very-high-dimensional datasets

Accuracy	LibSVM	RF-C4.5	RF-ODT	RF-ODT vs LibSVM	RF-ODT vs RF-C4.5
mean	87.28	90.07	93.63	**6.35**	**3.57**
standard deviation	13.65	22.31	21.69	9.41	6.32
student ratio				2.61	2.19
p-value				**0.0204**	**0.0462**
result of RF-ODT				gain*	gain*
RF-ODT win				10	9
RF-ODT tie				4	6
RF-ODT defeat				1	0
p-value				0.0117	0.0039
result of RF-ODT				gain*	gain**

Table 6 Precision comparison on very-high-dimensional datasets

Precision	LibSVM	RF-C4.5	RF-ODT	RF-ODT vs LibSVM	RF-ODT vs RF-C4.5
mean	84.62	89.74	92.73	8.11	2.99
standard deviation	15.30	14.69	10.38	9.26	7.68
student ratio				3.39	1.51
p-value				**0.0044**	0.1540
result of RF-ODT				gain**	
RF-ODT win				11	6
RF-ODT tie				3	6
RF-ODT defeat				1	3
p-value				**0.0063**	0.5078
result of RF-ODT				gain**	

Table 7 Recall comparison on very-high-dimensional datasets

Recall	LibSVM	RF-C4.5	RF-ODT	RF-ODT vs LibSVM	RF-ODT vs RF-C4.5
mean	86.80	83.06	90.17	3.37	7.11
standard deviation	18.37	24.94	14.13	6.98	11.37
student ratio				1.87	2.42
p-value				0.0828	**0.0296**
result of RF-ODT					gain*
RF-ODT win				6	8
RF-ODT tie				6	6
RF-ODT defeat				3	1
p-value				0.5078	0.0391
result of RF-ODT					gain*

Table 8 F1-measure comparison on very-high-dimensional datasets

F1-measure	LibSVM	RF-C4.5	RF-ODT	RF-ODT vs LibSVM	RF-ODT vs RF-C4.5
mean	84.67	84.54	90.84	**6.18**	**6.31**
standard deviation	15.61	27.15	22.65	6.90	9.88
student ratio				3.47	2.47
p-value				**0.0038**	**0.0269**
result of RF-ODT				gain**	gain*
RF-ODT win				12	10
RF-ODT tie				2	5
RF-ODT defeat				1	0
p-value				0.0034	0.0020
result of RF-ODT				gain**	gain**

The comparison dataset by dataset gives a very significant advantage to RF-ODT (sign-test p-value = 0.0020) which is never defeated by RF-C4.5. Indeed, RF-ODT is the winner 10 times out of 15 and there is equality 5 times out of 15.

The comparison of RF-ODT with LibSVM gives an opposite result. The superiority of RF-ODT on LibSVM is mainly due to the increase of the precision. In fact, RF-ODT improves the LibSVM precision of 8.1 percent points on average (table 6), which is very significant (p-value = 0.0044). Out of the 15 datasets, RF-ODT is beaten only once by LibSVM (11 wins, 3 ties, 1 defeat, p-value = 0.0063). RF-ODT improves the LibSVM recall of 3.4 percent points on average (table 7), which is not quite significant. However, we note that the 6 wins (especially on datasets 1, 4 and 15) are larger than the 3 defeats. Overall, the F1-measure is improved (table 8) by 6.2 points on average (p-value = 0.0038), which is very significant. The dataset by dataset comparison supports this conclusion (12 wins, 2 ties, 1 defeat, p-value = 0.0034).

3.3 Classification results on standard datasets

It is interesting to complement the above experiments by comparing the accuracies of RF-ODT and RF-C4.5 on standard benchmarks. Ten datasets (table 2) are used, each of them having a number of variables comprised between 6 and 64 and a ratio between the number of dimensions and the number of datapoints which does not exceed 5%.

These experiments (table 9) suggest that RF-ODT is at least as effective as RF-C4.5 when the datasets are standard. Indeed, RF-ODT improves the RF-C4.5 accuracy of 0.6 percent points on average, but this advantage is tiny and not significant. The differences between the accuracies of RF-ODT and RF-C4.5 on each dataset are small, except for Bupa dataset where RF-ODT ensures an increase of 4 percent points. However, it must be noticed that RF-ODT wins 8 times out of 10 against RF-C4.5, which is almost significant (p-value = 0.0547).

The whole experiments confirm the validity of our approach: RF-ODT at least match RF-C4.5 on standard datasets and outperforms RF-C4.5 on very high dimensional databases, which is the pursued aim.

3.4 Execution time

Before to compare the RF-ODT execution time and the RF-C4.5 one, let us analyse the theoretical complexity of the both algorithms. Considering m datapoints and n attributes, RF-C4.5 constructs the k trees of the forest with the complexity $O(kqn'mlog(m))$, where $q = 1$ if the attributes are nominal, $q = 2$ if the attributes are numerical and $n' << n$. Our RF-ODT algorithm has complexity $O(kp(n'+m)n'^2)$, p being the average depth of the different trees.

Table 9 Accuracy comparison of random forests on standard datasets

Accuracy	RF-ODT vs RF-C4.5
mean	**0.69**
standard deviation	3.10
student ratio	0.70
p-value	**0.5001**
result of RF-ODT	non significant
RF-ODT win	8
RF-ODT tie	0
RF-ODT defeat	2
p-value	0.1094
result of RF-ODT	almost significant

In practice, the size of oblique trees is smaller than the size of C4.5 ones. An oblique tree generally has a smaller depth. In addition, at each node of each RF-ODT tree, the subset of attributes is reduced. For these reasons, the RF-ODT execution time is faster than the RF-C4.5 one. The experiments to compare the RF-ODT and RF-C4.5 execution time were performed on a PC (Pentium 2,4 GHz, 1 Go RAM, Linux Mandriva 2008). Execution time is given in Table 10.

Table 10 Execution time

Dataset	Execution time			Dataset	Execution time		
ID	RF-C4.5	RF-ODT	RF-C4.5/RF-ODT	ID	RF-C4.5	RF-ODT	RF-C4.5/RF-ODT
1	2.96	**0.66**	4,48	14	**4.00**	4.30	0,93
2	3.96	**3.42**	1,16	15	**74.68**	86.20	0,87
3	**5.98**	17.97	0,33	16	**0.21**	0.28	0,75
4	11.38	**6.58**	1,73	17	1.87	**0.15**	12,47
5	7.41	**6.81**	1,09	18	0.93	**0.83**	1,12
6	3.09	**2.49**	1,24	19	1.87	**0.20**	9,35
7	**1.30**	2.50	0,52	20	10.08	3.65	2,76
8	**3.33**	5.88	0,57	21	2.79	**1.07**	2,61
9	797.10	**671.00**	1,19	22	7.83	**3.74**	2,09
10	29.97	**22.99**	1,30	23	4.90	**1.35**	3,63
11	2.55	**1.73**	1,47	24	8.62	**1.56**	5,53
12	19.82	**10.37**	1,91	25	28.07	**6.56**	4,28
13	17.70	**11.96**	1,48	Mean	42.10	**34.97**	1,20

In addition, we evaluated our proposal on Forest cover type is which a very large dataset [Asuncion and Newman, 2007]. It comprises 495141 datapoints for the learning set and 45141 datapoints for the test set. We constructed random forest of 30 trees to learn each of the two largest classes (Spruce-Fire: 211840 datapoints and Lorgepole-Pine: 283301 datapoints with 54 attributes). The learning time of RF-ODT is 801.61 seconds with a precision of 99.98%, while the learning time of RF-C4.5 is 17484 seconds with a precision of 99.57%. Therefore our RF-ODT is 22 times faster than the usual RF-C4.5, while it slightly improves the precision (0.41%).

4 Conclusion and future works

We presented random forests of oblique decision trees that achieve high performances for classification tasks. The main ideas are to use linear proximal SVMs for performing multivariate node splitting during tree construction, producing individual classifiers that are stronger than in a classical forests. Numerical test results on standard datasets and very-high-dimensional datasets have shown that our random forests of oblique decision trees algorithm is usually more accurate in terms of precision, recall, F1-measure, accuracy compared with random forests of C4.5 and SVM. It has significant better performance on very-high-dimensional data with better -but not significant- results on lower dimensional datasets. In addition our proposal is very efficient and it can be parallelized. A parallel implementation that exploits the multicore processors can greatly speed up the learning tasks.

Extension of the proposed approach for imbalanced datasets, multi-class classification, regression problems and feature selection tasks are under progress. In the near future we intend to provide more empirical test on large benchmarks and comparisons with other oblique trees methods.

References

[Amit and Geman, 2001] Amit, Y. and Geman, D. (2001). Shape quantization and recognition with randomized trees. *Machine Learning*, 45(1):5–32.

[Asuncion and Newman, 2007] Asuncion, A. and Newman, D. (2007). UCI repository of machine learning databases. http://www.ics.uci.edu/∼mlearn/MLRepository.html.

[Bennett and Mangasarian, 1994] Bennett, K. P. and Mangasarian, O. L. (1994). Multicategory discrimination via linear programming. *Optimization Methods and Software*, 3:27–39.

[Breiman, 1996] Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24(2):123–140.

[Breiman, 2001] Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

[Breiman et al., 1984] Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. (1984). *Classification and Regression Trees*. Wadsworth International,.

[Buntine, 1992] Buntine, W. (1992). Learning classification trees. *Statistics and Computing*, 2:63–73.

[Carvalho and Freitas, 2004] Carvalho, D. and Freitas, A. (2004). A hybrid decision tree/genetic algorithm method for data mining. *Information Sciences*, 163(1-3):13–35.

[Chang and Lin, 2001] Chang, C. C. and Lin, C. J. (2001). LIBSVM – a library for support vector machines. http://www.csie.ntu.edu.tw/∼cjlin/libsvm.

[Cohen et al., 2007] Cohen, S., Rokach, L., and Maimon, O. (2007). Decision-tree instance-space decomposition with grouped gain-ratio. *Information Sciences*, 177(17):3592–3612.

[Cutler and Guohua, 2001] Cutler, A. and Guohua, Z. (2001). PERT – perfect random tree ensembles. *Computing Science and Statistics*, 33:490–497.

[Dietterich, 2000a] Dietterich, T. G. (2000a). Ensemble methods in machine learning. In *Multiple Classifier Systems*, pages 1–15.

[Dietterich, 2000b] Dietterich, T. G. (2000b). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40(2):139–157.

[Do et al., 2009] Do, T.-N., Lallich, S., Pham, N.-K., and Lenca, P. (2009). Un nouvel algorithme de forêts aléatoires d'arbres obliques particulièrement adapté à la classification de données en

grandes dimensions. In Ganascia, J.-G. and Gançarski, P., editors, *Extraction et Gestion des Connaissances 2009*, pages 79–90, Strasbourg, France.

[Do and Poulet, 2006] Do, T. N. and Poulet, F. (2006). Classifying one billion data with a new distributed svm algorithm. In *Proceedings RIVF-2006: the 4th IEEE International Conference on Computer Science, Research, Innovation and Vision for the Future*, pages 59–66.

[Freund and Schapire, 1995] Freund, Y. and Schapire, R. (1995). A decision-theoretic generalization of on-line learning and an application to boosting. In *Computational Learning Theory: Proceedings of the Second European Conference*, pages 23–37.

[Fung and Mangasarian, 2001] Fung, G. and Mangasarian, O. (2001). Proximal support vector classifiers. In *Proceedings KDD-2001: Knowledge Discovery and Data Mining*, pages 77–86.

[Geurts et al., 2006] Geurts, P., Ernst, D., and Wehenkel, L. (2006). Extremely randomized trees. *Machine Learning*, 63(1):3–42.

[Heath, 1992] Heath, D. (1992). *A Geometric Framework for Machine Learning*. PhD thesis, Johns Hopkins University, Baltimore, Maryland.

[Ho, 1995] Ho, T. K. (1995). Random decision forest. In *Proceedings of the Third International Conference on Document Analysis and Recognition*, pages 278–282.

[Jinyan and Huiqing, 2002] Jinyan, L. and Huiqing, L. (2002). Kent ridge bio-medical data set repository. Technical report. http://datam.i2r.a-star.edu.sg/datasets/krbd/.

[Loh and Vanichsetakul, 1988] Loh, W.-Y. and Vanichsetakul, N. (1988). Tree-structured classification via generalized discriminant analysis (with discussion). *Journal of the American Statistical Association*, 83:715–728.

[Maji, 2008] Maji, P. (2008). Efficient design of neural network tree using a new splitting criterion. *Neurocomputing*, 71(4-6):787–800.

[Michie et al., 1994] Michie, D., Spiegelhalter, D. J., and Taylor, C. C., editors (1994). *Machine Learning, Neural and Statistical Classification*. Ellis Horwood.

[Murthy et al., 1994] Murthy, S., Kasif, S., and Salzberg, S. (1994). A system for induction of oblique decision trees. *Journal of Artificial Intelligence Research*, 2(1):1–32.

[Murthy et al., 1993] Murthy, S., Kasif, S., Salzberg, S., and Beigel, R. (1993). OC1: Randomized induction of oblique decision trees. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 322–327.

[Quinlan, 1993] Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA.

[Robnik-Sikonja, 2004] Robnik-Sikonja, M. (2004). Improving random forests. In *Proceedings of the Fifth European Conference on Machine Learning*, pages 359–370.

[Rokach and Maimon, 2005] Rokach, L. and Maimon, O. (2005). Top-down induction of decision trees classifiers - a survey. *IEEE Transactions on Systems, Man and Cybernetics. Part C: Applications and Reviews*, 35(4):476–487.

[Simon et al., 2009] Simon, C., Meessen, J., and De Vleeschouwer, C. (2009). Embedding proximal support vectors into randomized trees. In *European Symposium on Artificial Neural Networks, Advances in Computational Intelligence and Learning*, pages 373–378.

[Suykens and Vandewalle, 1999] Suykens, J. and Vandewalle, J. (1999). Least squares support vector machines classifiers. *Neural Processing Letters*, 9(3):293–300.

[van Rijsbergen, 1979] van Rijsbergen, C. V. (1979). *Information Retrieval*. Butterworth.

[Vapnik, 1995] Vapnik, V. (1995). *The Nature of Statistical Learning Theory*. Springer-Verlag.

[Veropoulos et al., 1999] Veropoulos, K., Campbell, C., and Cristianini, N. (1999). Controlling the sensitivity of support vector machines. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 55–60.

[Wolpert, 1992] Wolpert, D. (1992). Stacked generalization. *Neural Networks*, 5:241–259.

[Wu et al., 1999] Wu, W., Bennett, K., Cristianini, N., and Shawe-Taylor, J. (1999). Large margin trees for induction and transduction. In *Proceedings of the Sixth International Conference on Machine Learning*, pages 474–483.

[Xu et al., 2006] Xu, Q., Pei, W., Yang, L., and He, Z. (2006). Support vector machine tree based on feature selection. In King, I., Wang, J., Chan, L., and Wang, D. L., editors, *13th International Conference Neural Information Processing*, volume 4232 of *Lecture Notes in Computer Science*, pages 856–863, Hong Kong, China. Springer.

[Yildiz and Alpaydin, 2005] Yildiz, O. and Alpaydin, E. (2005). Linear discriminant trees. *International Journal of Pattern Recognition and Artificial Intelligence*, 19(3):323–353.

[Zhou and Chen, 2002] Zhou, Z.-H. and Chen, Z.-Q. (2002). Hybrid decision tree. *Knowledge-Based Systems*, 15(8):515–528.