ACKNOWLEDGMENT

The authors would like to thank two anonymous reviewers for their insightful comments and valuable suggestions on the earlier versions of this letter.

REFERENCES

[1] S. Haykin, *Neural Networks*. Upper Saddle River, NJ: Prentice-Hall, 1999.
[2] "A normalised real time recurrent learning algorithm," *Elsevier Signal Processing*, vol. 80, no. 9, pp. 1909–1916, 2000.
[3] D. P. Mandic and J. A. Chambers, *Recurrent Neural Networks for Prediction: Learning Algorithms, Architecture and Stability*. Chichester, U.K.: Wiley, 2001.
[4] Oliver and Nelles, *Nonlinear System Identification*. New York: Springer-Verlag, 2001.
[5] F. C. Sun, Z. Q. Sun, and P. Y. Woo, "Neural network-based adaptive controller design of robotic manipulators with an observer," *IEEE Trans. Neural Netw.*, vol. 12, no. 1, pp. 54–67, Jan. 2001.
[6] R. Williams and J. Peng, "Gradient-based learning algorithms for recurrent neural networks," *Neural Comput.*, vol. 2, pp. 490–501, 1990.
[7] M. W. Mak, K. W. Ku, and Y. L. Lu, "On the improvement of the real time recurrent learning algorithm for recurrent neural networks," *Neurocomputing*, vol. 24, pp. 13–36, Feb. 1999.
[8] Q. Song, J. Xiao, and Y. C. Soh, "Robust back-propagation training algorithm for multi-layered neural tracking controller," *IEEE Trans. Neural Netw.*, vol. 10, no. 5, pp. 1133–1141, Sep. 1999.
[9] Y. L. Wu, Q. Song, and X. L. Yang, "Robust recurrent neural network control of biped robot," *J. Intell. Robot. Syst.*, vol. 49, no. 2, pp. 151–169, Jun. 2007.
[10] M. Rupp and A. H. Sayed, "A time-domain feedback analysis of filtered-error adaptive gradient algorithms," *IEEE Trans. Signal Process.*, vol. 44, no. 6, pp. 1428–1439, Jun. 1996.
[11] M. RuppAli and H. Sayed, "Supervised learning of perceptron and output feedback dynamic networks: a feedback analysis via the small gain theorem," *IEEE Trans. Neural Netw.*, vol. 8, no. 3, pp. 612–622, May 1997.
[12] D. P. Mandic, "Data-reusing recurrent neural adaptive filters," *Neural Comput.*, vol. 14, pp. 2693–2707, 2002.
[13] A. D. Back, "Locally recurrent globally feedforward networks: A critical review of architectures," *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 229–239, Mar. 1994.
[14] R. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Comput.*, vol. 1, pp. 270–280, 1989.
[15] A. E. Nordsjo and L. H. Zetterberg, "Identification of certain time-varying nonlinear Wiener and Hammerstein systems," *IEEE Trans. Signal Process.*, vol. 49, no. 3, pp. 577–592, Mar. 2001.

Cline: A New Decision-Tree Family

M. Fatih Amasyalı and Okan Ersoy

***Abstract*—A new family of algorithm called Cline that provides a number of methods to construct and use multivariate decision trees is presented. We report experimental results for two types of data: synthetic data to visualize the behavior of the algorithms and publicly available eight data sets. The new methods have been tested against 23 other decision-tree construction algorithms based on benchmark data sets. Empirical results indicate that our approach achieves better classification accuracy compared to other algorithms.**

***Index Terms*—Classifier combination, decision forests, decision trees, learning (artificial intelligence), machine learning, multivariate decision trees, pattern classification, pattern recognition.**

I. INTRODUCTION

Decision-tree algorithms are among the most used tools in pattern recognition applications. A typical decision-tree algorithm begins with all the data, splits the data into two subsets based on the values of one or more attributes on decision nodes, and then recursively splits each subset into finer ones, until each subset contains a single class. These final subsets form the leaf nodes of the decision tree. The classification process starts at the root node. The data follows the directions of the decision nodes until it reaches a leaf. Upon reaching a leaf, the class of the data is assigned as the class of the leaf. Decision trees may be univariate, linear multivariate, or nonlinear multivariate depending on whether a single attribute, a linear function of all the attributes, or a nonlinear function of all the attributes is used for the partitioning at each node of the decision tree [13]. If the tree is univariate, its decision nodes hold only one feature and one threshold value. The data coming into a particular node is directed to a particular branch according to whether the feature of the data on this node is greater than the threshold value. Multivariate trees have hyperplanes in their nodes. Each hyperplane $(w_{1..p+1})$ consists of a linear combination of all features [1], [14], and implements (1). In (1), each data sample has p features

$$f(x) : w_m^T x + w_{m0} = \sum_{j=1}^{p} w_{mj} x_j + w_{m0} > 0. \tag{1}$$

An incoming data of a node is directed to a particular branch according to which part of the hyperplane the data is on. Univariate and multivariate decision nodes which are constructed for the same data set are illustrated in Fig. 1. In Fig. 1, $u1$, $u2$, and $u3$ univariate splitters are needed for a decision which can be carried out by only one multivariate splitter $(m1)$ shown. Many studies show that the multivariate trees are generally smaller and more accurate but less understandable [2].

To be able to compute the weighted sum in (1), all the features should be numeric and discrete or nonordered values need be represented numerically (usually by 1-of-L encoding) beforehand [1].

Manuscript received October 14, 2006; revised February 18, 2007 and June 22, 2007; accepted July 2, 2007. An early version of this work was presented at the ICSC Congress on Computational Intelligence Methods and Applications (CIMA'05), Istanbul, Turkey.

M. F. Amasyalı is with the Computer Engineering Department, Yıldız Technical University, Istanbul 34349, Turkey (e-mail: mfatih@ce.yildiz.edu.tr).

O. Ersoy is with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907 USA (e-mail: ersoy@purdue.edu).

Color versions of one or more of the figures in this letter are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNN.2007.910729

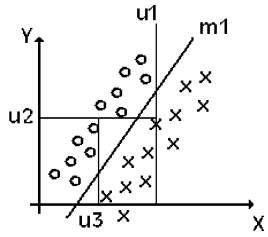

Fig. 1. Univariate and multivariate decision nodes.

When building a univariate decision tree, all the features and threshold values can be tested, and the most suitable one can be chosen. However, this method is impractical for multivariate trees due to the huge number of possibilities. To overcome this problem, many methods have been developed in the literature. This letter presents a new approach for this problem as well.

In this letter, we present a new algorithm family named Cline, incorporating a number of different methods for building multivariate decision trees, and compare them with 23 other decision-tree construction methods. The Cline family of algorithms are based on some heuristic methods to determine hyperplanes, and they are different from previous methods in a number of ways. In the literature, previous hyperplane determination algorithms such as the perceptrons and support vector machines (SVMs) usually try to optimize a function. The Cline family of algorithms does not have this property. The learning vector quantization (LVQ) algorithm tries to find the best representative centroids of classes. Its first usage in decision trees is CLLVQ (see Section II-A for definition), which is an algorithm in the Cline family [12]. In linear discriminant analysis (LDA), one tries to find a transformation of data set which leads to separability of classes as much as possible. LDA is used in several works for multivariate decision trees [11]. In Alpaydin's works, the separation plane is found using LDA in which the threshold value is estimated assuming that the classes are normally distributed with equal variances. In comparison, in the Cline family, it is found with a heuristic approach. Previous decision-tree algorithms typically use the same algorithm in all nodes. The Cline family algorithm to be referred to as CLMIX (see Section II-A for definition) differs from others. It tries several algorithms at each node and uses the best one. In this way, it searches for the best fitting algorithm at each node/current subspace.

This letter consists of four sections. The Cline algorithms are discussed in Section II. The comparison of the Cline and other methods on eight different benchmark data sets is discussed in Section III. The conclusions and future works are given in Section IV.

II. Family of Cline Algorithms

The decision trees constructed with the Cline algorithms are multivariate trees. At each node, all of the features or a subset of the features of instances are used for decision making. A Cline tree is also a binary tree because two branches are generated from each internal node. It can be used for all classification problems based on numerical values.

Cline algorithms generate nodes which divide the feature space into two regions with a hyperplane. The determinations of the hyperplanes are discussed in Section II-A. The division process is repeated until each region contains single-class data. In the 2-D space, the boundary which separates classes is a line, in 3-D space, it is a plane, and in higher dimensional spaces, it is a hyperplane. At each node, there is a hyperplane and the test instances are directed within the tree according to whether the test point is on one side, or the other side of the hyperplane.

A. Determination of the Boundary Hyperplanes

We developed six heuristic methods (CL2, CL4, CLM, CLLDA, CLLVQ, CLMIX) to determine the boundary hyperplanes. The methods will be explained in 2-D space for the purpose of easy visualization. In 2-D space, classes are separated with a line. Fig. 2 illustrates 10 instances and their binary classification with Cline algorithms.

CL2 (Cline2point): In Fig. 2(a), the nearest two points which are from different classes are "A" and "B," which are determined by

$$(i,j) = \arg\min_{i,j} \{\mathrm{dist}(x_i, x_j) | x_i \in C_1, x_j \in C_2\}, \quad A = x_i, \qquad B = x_j. \tag{2}$$

Next, the midpoint of the A–B line segment is found by

$$m = \frac{1}{2}\sum_{j=1}^{p} A_j + B_j. \tag{3}$$

There is only one line defined by the parameter vector w which passes through the midpoint of the A–B line segment (w') and perpendicular to it. w is determined by.

$$\sum_{j=1}^{p} w_j' A_j + w_0' = \sum_{j=1}^{p} w_j' B_j + w_0' = 0$$
$$w' \times w = -1$$
$$\sum_{j=1}^{p} w_j m_j + w_0 = 0. \tag{4}$$

This line defined by w is used as the classification line in the CL2 algorithm.

CL4 (Cline4point): "A" and "B" points are found using (2) as in the CL2. The "D" point is the second nearest point (which is from the other class) to the "A" point after the "B" point. Similarly, the "C" point is the second nearest point to the "B" point after the "A" point. The points "C" and "D" are computed by

$$k = \arg\min_{k} \{\mathrm{dist}(x_k, B) | x_k \in C_1, x_k \neq A\}, \qquad C = x_k \tag{5}$$
$$t = \arg\min_{t} \{\mathrm{dist}(x_t, A) | x_t \in C_2, x_t \neq B\}, \qquad D = x_t. \tag{6}$$

There is only one line (line 2) which passes through the midpoint of the A–C line $(m1)$ and the midpoint of the B–D line $(m2)$. Let the parameters of this line be denoted by w'. $m1$, $m2$, and w' are determined by

$$m1 = \frac{1}{2}\sum_{j=1}^{p} A_j + C_j$$
$$m2 = \frac{1}{2}\sum_{j=1}^{p} B_j + D_j \tag{7}$$
$$\sum_{j=1}^{p} w_j' m1_j + w_0' = \sum_{j=1}^{p} w_j' m2_j + w_0' = 0. \tag{8}$$

Then, the line defined by the parameter vector w, perpendicular to line 2 and passing through its midpoint m' as shown in Fig. 2(b) is used as the classification line in the CL4 algorithm. w is determined by

$$m' = \frac{1}{2}\sum_{j=1}^{p} m1_j + m2_j$$
$$w' \times w = -1$$
$$\sum_{j=1}^{p} w_j m_j' + w_0 = 0. \tag{9}$$

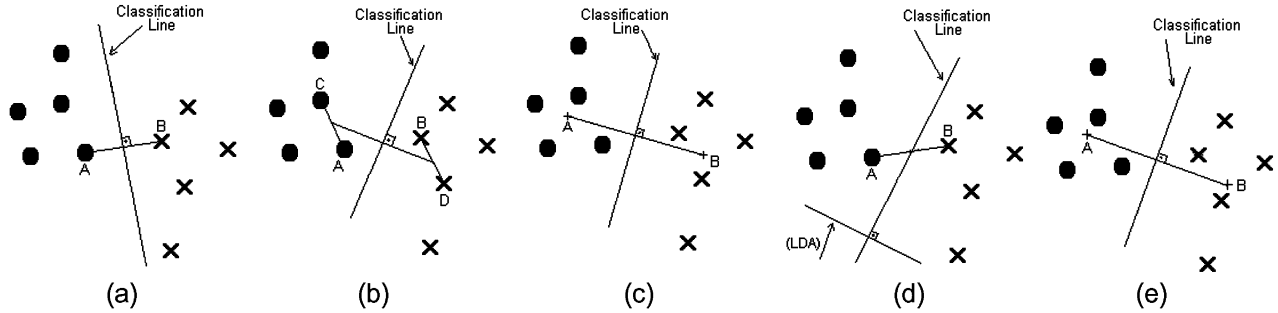

Fig. 2. Finding the classification line in (a) CL2, (b) CL4, (c) CLM, (d) CLLDA, and (e) CLLVQ.

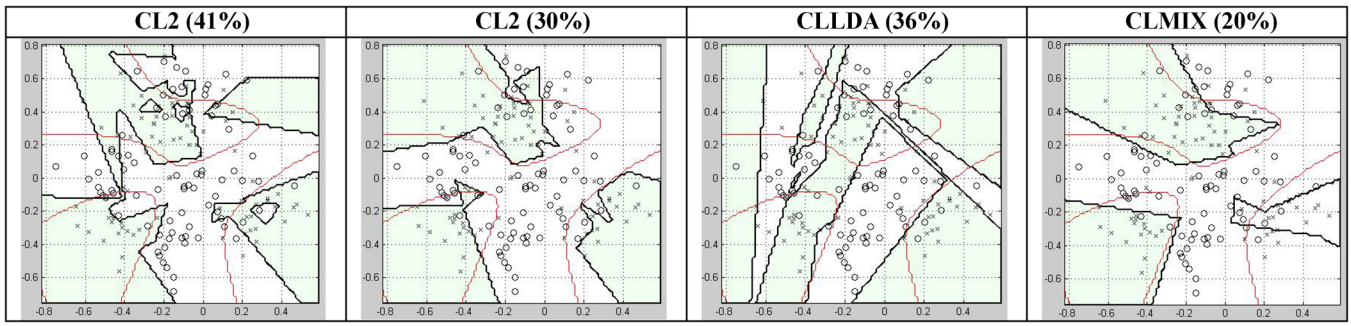

Fig 3. Decision boundaries and test errors of four Cline algorithms when two classes overlap substantially.

The tree construction speed is fast with the CL2 and CL4 versions. However, they are sensitive to outliers/noises. To overcome this problem the following versions were developed.

CLM (Cline Mean): In Fig. 2(c), the "A" and "B" points are the mean points of the classes. They are determined by

$$A = \{\text{mean}(x_i)|x_i \in C_1\} \quad B = \{\text{mean}(x_i)|x_i \in C_2\}. \tag{10}$$

There is only one line defined by the parameter vector w which passes through the midpoint m of the A–B line segment defined by the parameter vector w' and perpendicular to it. This line defined by the parameter vector w is referred to as the classification line and used in the CLM algorithm. The parameter vector w is found by using (4) as in the CL2 algorithm.

CLLDA (Cline Linear Discriminant Analysis): LDA [3] is used to transform the multidimensional data into 1-D data such that the distance between class centroids is maximized while the variances of the classes are minimized.

In Fig. 2(d), the nearest two points which are from different classes are "A" and "B." They are found using (2) as in the CL2 algorithm.

The midpoint m of the A–B line segment is found using (3). There is only one line defined by the parameter vector w which passes through the midpoint m of the A–B line defined by the parameter vector w' and perpendicular to the LDA's transformation hyperplane. LDA's transformation hyperplane determined by the parameter vector $w''_{1..p}$ maximizes the separability of the C1 and C2 classes when the input vector X is projected by using it. The parameter vector $w''_{1..p}$ is found by

$$w''_{1..p} = \text{LDA}(X, T) \tag{11}$$

where T is the class labels of samples. The classification line defined by the parameter vector w which is used in the CLLDA algorithm is determined by

$$w'' \times w = -1 \quad \sum_{j=1}^{p} w_j m_j + w_0 = 0. \tag{12}$$

TABLE I
SUMMARY OF DATA SETS

Dataset Name	Dataset Code	# of Features	# of Classes	# of Samples	Prunity Success (%)
Breast cancer Wisconsin	Bcw	9	2	683	64.71
Boston housing	Bos	12	3	506	33.33
Congressional voting	Vot	16	2	435	61.36
Bupa liver disorders	Bld	6	2	345	57.14
StatLog heart disease	Hea	7	2	270	55.56
Pima Indians diabetes	Pid	7	2	532	66.67
StatLog image segmentation	Seg	19	7	2310	14.29
StatLog vehicle silhouette	Veh	18	4	3772	25.88

CLLVQ (Cline Linear Vector Quantization): Using linear vector quantization (LVQ) [4], two best representative class centroids (A and B) are found in Fig. 2(e) by

$$[A, B] = \text{LVQ}(X, T). \tag{13}$$

There is only one line determined by the parameter vector w which passes through the midpoint m of the A–B line segment determined by the parameter vector w' and perpendicular to it. This line is used as the classification line in the CLLVQ algorithm. Its parameter vector w is determined by (4).

CLMIX (Cline Mix): While analyzing the classification accuracies of the Cline algorithms, it was observed that there is no single Cline version that performs better than others on all data sets. The best three versions are CLM, CLLDA, and CLLVQ. By combining these versions into one tree, the CLMIX version was developed. In CLMIX, at each node, the data points are classified using three methods one by one. The algorithm having best classification accuracy is used in the tree. The parameter vector for this algorithm is determined by

$$w_{\text{CLMIX}} = \underset{w \in \{w_{\text{CLM}}, w_{\text{CLLDA}}, w_{\text{CLLVQ}}\}}{\arg\max} cs(X, T, w) \tag{14}$$

TABLE II
DECISION-TREE CONSTRUCTION ALGORITHMS AND THEIR ABBREVIATIONS

Abbr.	Algorithm	Abbr.	Algorithm
QU0	QUEST versions (Loh, Shih 1997)	IB	IND versions (Buntine 1992)
QU1		IB0	
QL0		IM	
QL1		IM0	
FTU	FACT versions (Loh, Vanichsetakul 1998)	IC0	CART versions (Breiman, Freidman, Olshen, Stone 1984)
FTL		IC1	
C4T	C4.5 versions (Quinlan 1993)	ST0	S-PLUS versions (Clark, Pregibon 1993)
C4R		ST1	
OCU	OC1 versions (Murthy, Kasif, Salzberg 1994)	LMT	LMDT (Brodley, Utgoff 1995)
OCL		CAL	CAL5 (Müller, Wysotzki 1997)
OCM		T1	T1 single split (Holte 1993)

where $cs(X, T, w)$ calculates the true classification accuracy of the current data set (X) with T labels using the hyperplane with the parameter vector w.

In Fig. 2, all Cline algorithms classify the data perfectly. However, if the classes are nonseparable, different Clines algorithms behave differently. For visualization of this difference, a new highly overlapped artificial data set was created. In this case, the real class distributions were known, and the optimal Bayes error was 21%. The 60% of data was used for training and the rest was used for testing. In Fig. 3, the decision boundaries for four Cline algorithms can be seen. Bold and thin lines are for the Cline algorithm decision boundaries and Bayes decision boundaries, respectively. The test errors are also given above each Cline version.

The CL2 algorithm is very sensitive to noise. This can be seen from its complex decision boundaries and bad classification performance. The CLM algorithm is more robust than the CL2 algorithm, and its decision boundaries are less complex. The CLLDA algorithm is badly affected by noise as in the CL2 algorithm. The CLMIX algorithm creates the least complex boundaries and has the best classification accuracy with the smallest tree. This shows its high generalization ability. The test error (20%) of the CLMIX algorithm was very comparable to the optimal Bayes error (21%).

B. Types of Tree Classification

In a decision tree, we find the leaf node, following the decision branches starting from the root. At this point, the leaf node may contain a single class, and then the classification result is that class. On the other hand, the leaf node may contain points which belong to several classes in a pruned tree. In this case, we use the majority class as the classification result.

In this letter, we implement a new labeling technique. During the tree construction process, each node's class probability distribution is recorded. During testing, the class probability distributions which are at the test sample's path on the tree are summed. The test sample's class is chosen as the class having highest probability. This technique helps with overfitting. It takes into account a larger part of the tree. Using this technique, a subsequent pruning stage is not needed. In the rest of this letter, the conventional leaf labeling technique is called "using leaves," and our proposed technique is called "using branches."

C. Conversion of a Binary Classifier to an N-Class Classifier

The original Cline algorithm is developed for the two-class problem. There are two common methods to convert this type of classifier (Cline, SVM, etc.) to an N-class classifier [5]: "one versus all" and "one versus one." The second method is used more often in the literature because of its better performance. Our experiments with the Cline algorithms showed the same result; so, in Cline family, "one versus one" method was used in the experiments. In this method, we construct $N * (N - 1)/2$ binary classifiers, each to distinguish one class from another class. For each classifier, a subset of the training set including only the instances from the two classes is used. The test samples are classified with N binary classifiers and the test sample's class is assigned as the most voted class in the N classifiers' results.

TABLE III
AVERAGE ACCURACY RATES AND VARIANCES OF CLINE METHODS

Methods	Success Ratio
Using leaves	79.3±1.9
Using branches	79.9±2.1
Full trees	79.5±1.4
Pruned trees	79.7±2.5
CL2	77.4±0.6
CL4	77.6±1.1
CLLDA	80.8±1.5
CLLVQ	80.0±0.4
CLM	79.9±0.8
CLMIX	82.1±1.7

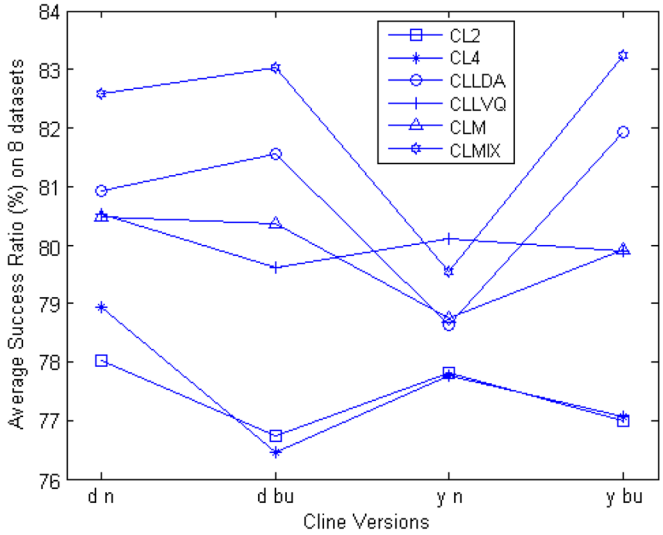

Fig. 4. Comparison of hyperplane methods.

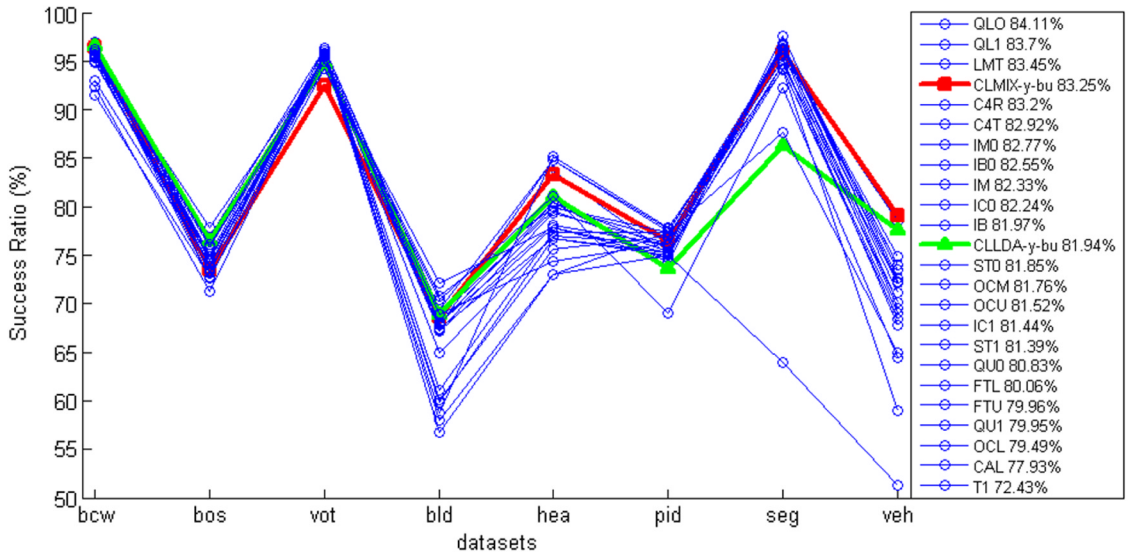

Fig. 5. Accuracy comparisons of single decision-tree construction methods and two Cline versions.

D. Decision Forests

Recently, the random forest algorithm [6] became very popular because of its very high accuracy with many data sets. The basic idea of the algorithm is to construct a decision forest instead of a single tree. The forest consists of several different multivariate decision trees which are trained by different training sets which are constructed from the original training set by bootstrap and random feature selection. Although each tree has its own decision, the maximum voted class in the forest is accepted as the final decision. Using the same idea, the forest versions of the Cline methods were developed and included in the Cline algorithm family. The basic difference between the Cline forest and the random forest is the decision-tree construction algorithm. The Cline forest algorithms use Cline methods, while random forest uses classification and regression tree (CART) [7] algorithm. In the rest of this letter, the term "decision tree" means a single decision tree while "decision forest" means a set of decision trees.

In random forest algorithm, the training sets of each tree are generated by bootstrapping (Breiman's bagging [6]) method which produce different replicates of original training set. The original training set and all replicates have the same number of samples. Replicates are generated by randomly selecting at random and with replacement examples from the original training set. This means that the same example may appear multiple times in a replicate, or it may not appear at all. This procedure is also added to the Cline forest algorithm.

III. Experimental Results

The experiments are reported in two sections. In Sections III-A and III-B, the experiments with the decision trees and the decision forests are described, respectively.

A. Decision-Tree Experiments

Lim *et al.* compared 33 new and old classification algorithms on 16 benchmark data sets [8]. The 22 of 33 algorithms were decision-tree algorithms. The success ratios were obtained using tenfold cross validation on 16 chosen data sets while a single test set was used on the remaining data sets. We used randomly selected eight data sets from the cross-validated data sets for comparison. We compared our results to Lim's classification accuracy results with these decision-tree algorithms. The detailed descriptions of 22 tree construction algorithms are given in [8]. The additional eight University of California at Irvine (UCI) data sets used [9] are shown in Table I with their names and characteristics. Prunity success is the percent success ratio if all the samples are classified as the most frequent class.

The names and abbreviations of algorithms discussed in [8] are given in Table II.

In this letter, $24(2*2*6)$ different Cline versions were run on the eight data sets by generating six different methods for determining hyperplanes, two methods for types of tree classification, and two types of trees (pruned and unpruned). The average accuracies and standard deviations of Cline methods are given in Table III.

It can be seen that the pruning process has little improvement on the average success ratios. Among hyperplane methods, the CLMIX has the highest accuracy. Among the types of tree classification, using branches has slightly higher accuracy than using leaves.

Six different hyperplane methods were compared in Fig. 4. In this figure, "d n" corresponds to using branch and no pruning, "d bu" corresponds to using branch and bottom-up pruning, "y n" corresponds to using leaves and no pruning, and "y bu" corresponds to using leaves and bottom-up pruning.

It is seen in Fig. 4 that the pruning process increases the success ratio except for CL2, CL4, and CLLVQ methods. The use of branches has higher accuracy than the use of leaves when the trees are not pruned. The branches method can be applied with other decision-tree algorithms as well.

In Fig. 5, 22 tree construction algorithms are compared with two Cline versions on the eight data sets, where X-axis shows data set's abbreviations while Y-axis shows the accuracy rates.

The best performed decision-tree construction system in [8] was quick unbiased efficient statistical tree (QUEST) with 84.11% accuracy. This method is a revised version of fast algorithm for classification trees (FACT). FACT's decision nodes have K branches for the data sets having K classes. Each decision node includes hyperplanes obtained using LDA assuming equal variances. In the QUEST algorithm, the decision nodes have two branches at each node. They group the classes into two super classes before an application of quadratic discriminant analysis (QDA) which does not assume equal variances [10].

Cline's best tree is the fourth most successful algorithm with 83.25% accuracy. It is observed that Cline versions' accuracies are competitive with the best algorithms.

TABLE IV
AVERAGE ACCURACIES AND STANDARD DEVIATIONS OF CLINE FOREST AND RANDOM FOREST VERSIONS

		CLM Forest	CLLDA Forest	CLLVQ Forest	CLMIX Forest	CLMIX ForestBS	Random Forest
Number of trees	10	81.29 ±0.87	82.89 ±1.26	81.37 ±1.15	82.65 ±1.23	81.93 ±1.77	82.80 ±1.13
	20	82.34 ±0.74	82.53 ±1.35	82.18 ±0.52	83.03 ±1.21	83.16 ±1.6	83.95 ±1.06
	30	82.27 ±0.4	82.39 ±1.74	82.34 ±0.43	83.56 ±1.41	83.64 ±1.35	84.09 ±1.05
	100	82.82 ±0.27	82.52 ±1.43	82.85 ±0.25	83.76 ±0.83	84.64 ±1.03	84.23 ±1.03
Number of features in decision nodes	$\log_2 M$	81.84 ±0.91	81.46 ±0.88	81.75 ±0.93	82.38 ±0.71	82.13 ±1.29	84.01 ±1.05
	$2\log_2 M$	82.52 ±0.51	83.70 ±0.26	82.62 ±0.37	84.12 ±0.79	84.56 ±0.9	83.52 ±1.06
Type of tree classification	Leaves	81.90 ±0.75	82.49 ±1.38	82.12 ±1.01	82.85 ±0.96	83.22 ±1.78	---
	Branches	82.46 ±0.79	82.67 ±1.33	82.25 ±0.63	83.65 ±1.26	83.47 ±1.62	

B. Decision Forest Experiments

The performance of decision forests instead of a single decision tree was evaluated with the same eight data sets. The forest versions of four Cline best single-tree construction algorithms were developed and compared with the random forest. The constructed trees were not pruned in the random forest. The pruning process was not applied in the Cline forest either for fair comparison of the algorithms. As a result, 80 [(four hyperplane methods + one bootstrapping) $*$ four different numbers of trees in forest $*$ two different numbers of features $*$ two different types of tree classification] different Cline forest versions, eight (four different numbers of trees $*$ two different numbers of features) different Random forest versions were run on eight data sets using tenfold cross validation. The average success ratios are given in Table IV. The M value represents the original feature number in Tables IV and V.

In the Cline forest and random forest versions, increasing the number of trees in the forest causes more successful classifiers and less standard deviations. This observation is parallel to Breiman's work who states that increasing the number of trees does not cause overfitting [6]. The Cline forest versions give better results and less standard deviations when the number of features in the trees is increased, while the random forest versions give worse results and more standard deviations. Using branches has higher accuracies than using leaves, which is similar to what was observed with Cline's single-tree construction versions. There is no meaningful differences in standard deviations between usage of branches and leaves. The bootstrapping procedure is added to CLMIX forests. It generally gives more successful results.

The accuracy rates and standard deviations of four Cline forest, four random forest versions and 22 tree construction algorithms are shown in Table V. The algorithms are sorted according to mean accuracy rates. The explanations of decision-tree algorithms can be seen in Table II. The method names of decision forests (random and Cline) consist of the hyperplane method, the number of trees, the usage of leaf or branches, and the number of features in each tree. Table V also includes the total number of "$*$" and "?" marks for each algorithm. The "$*$" mark indicates that the algorithm has a success rate within one standard deviation of the maximum for the data set. The "?" indicates that the algorithm has the worst success rate for the data set.

In Table V, the algorithms are sorted by mean accuracy rates. The four most successful algorithms belong to Cline family. Cline forest algorithms give better result on "bcw" and "hea" data sets whereas Random forest has higher results on "bld" and "seg" data sets. It is also observed that decision forests give better results than decision trees. The best single decision-tree algorithm is QUEST with 84.11% accuracy while the best forest is Cline forest (bootstrapped) constructed with 100 trees having 85.65% accuracy, as seen in Table V.

The four most successful algorithms over all eight data sets also belong to the Cline family. This was expected since the random forest uses the CART algorithm to determine boundary hyperplanes, and the Cline algorithms (especially CLMIX) are more successful than the CART (IC0, IC1) algorithms in single-tree construction algorithms (see Fig. 5). It can be reasoned that combining more accurate decision trees probably generates more accurate decision forests.

The algorithms are also compared according to their training times. In Fig. 6, the axes are mean success rate and mean training time on eight data sets. Mean training time axis is logarithmic whereas success rate is linear. The time measure is in seconds.

The best performance algorithm CLineMIX forest with 100 trees (CLMIXBS100) takes more time than random forests. There is also a positive correlation between algorithm complexity (training time) and performance. However, the most complex algorithm CAL is not the best performing algorithm. There are also several other algorithms which take more time than Cline forests but are less accurate.

IV. CONCLUSION AND FUTURE WORK

The performances of the proposed methods were compared with 23 other decision-tree algorithms based on eight data sets. Six new decision-tree methods were developed. The new decision-tree methods were shown to be competitive with other decision-tree methods in terms of performance. The single-tree Cline algorithms were ranked fourth among 22 algorithms. The most successful method we developed is CLMIX which uses several constructing methods to determine the hyperplanes in a single tree.

When classifying data in traditional decision-tree algorithms, only the labels of tree leaves are typically used. In this letter, not only the class distributions of the leaves but also the class distributions of all the nodes through which the data passes during their travel within the tree are also considered. Better classification results were obtained with this approach.

Instead of building a single tree, Breiman combines the results of many different trees that are trained with different data sets in the "random forest" method. We also used this method for Cline and developed the forest versions of the Cline methods. After the experiments on eight data sets, it was concluded that building a forest instead of a single tree improves performance. The number of trees in a forest and the number of features of each tree in the forest were analyzed on random

TABLE V
SUCCESS COMPARISON OF THE BEST FOUR CLINE FOREST AND FOUR RANDOM FOREST WITH 22 TREE CONSTRUCTION ALGORITHMS FOR EACH DATA SET. THE MEAN ACCURACY RATES AND THEIR MEAN STANDARD DEVIATIONS ARE GIVEN IN THE TENTH AND ELEVENTH COLUMNS. "$*$" MARK INDICATES THAT THE ALGORITHM HAS A SUCCESS RATE WITHIN ONE STANDARD DEVIATION OF MAXIMUM FOR THE DATA SET. "?" INDICATES THAT THE ALGORITHM HAS THE WORST SUCCESS RATE FOR THE DATA SET

	Datasets								Mean	Mean		
Methods	bcw	bos	vot	bld	hea	pid	seg	veh	acc	std	# of *	# of ?
CLMIX_forestBS_100_y_2log$_2$M	*	*		*	*	*	*	*	85.65	3.50	7	0
CLMIX_forestBS_100_d_2log$_2$M	*			*	*	*	*	*	85.42	3.53	6	0
CLMIX_forestBS_30_d_2log$_2$M	*	*		*	*	*	*	*	84.89	3.93	7	0
CLMIX_forestBS_30_y_2log$_2$M	*	*		*	*	*	*	*	84.69	3.73	7	0
Random Forest_100 _log$_2$M	*	*		*	*	*	*	*	84.54	4.43	7	0
Random Forest_30_ log$_2$M	*	*		*	*		*	*	84.33	4.41	6	0
QLO	*		*		*	*	*	*	84.11		6	0
Random Forest_20_ log$_2$M	*	*	*	*			*	*	84.10	4.30	6	0
Random Forest_100_ 2log$_2$M		*		*		*	*	*	83.92	4.25	5	0
QL1	*				*	*	*	*	83.70		5	0
LMT	*				*		*	*	83.45		4	0
C4R	*			*		*	*		83.21		4	0
C4T		*					*		82.92		2	0
IM0	*		*				*		82.77		3	0
IB0	*		*				*		82.55		3	0
IM						*	*		82.33		2	0
IC0							*		82.24		1	0
IB							*		81.97		1	0
ST0							*	*	81.85		2	0
OCM			?	*			*		81.76		2	1
OCU							*		81.52		1	0
IC1							*		81.44		1	0
ST1							*		81.39		1	0
QU0	*		*			*	*		80.83		4	0
FTL					*	*	*		80.06		3	0
FTU							*		79.96		1	0
QU1						*	*		79.95		2	0
OCL	?			*		?	*		79.49		2	2
CAL					?	*			77.93		1	1
T1		?		?	?		?	?	72.43		0	5

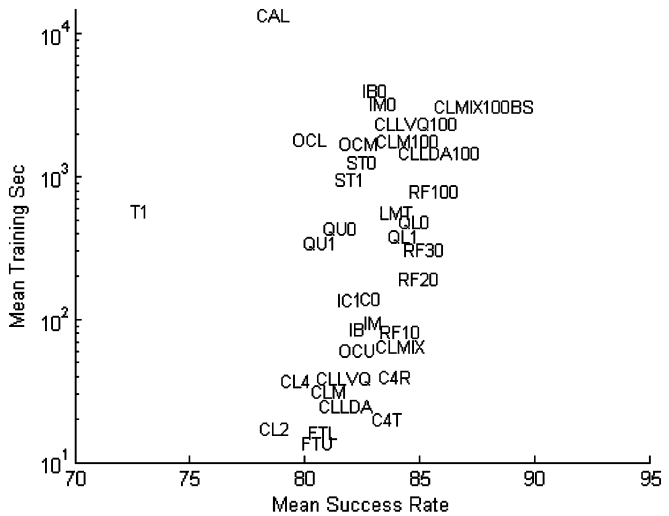

Fig. 6. Comparison of training times and success ratios of 36 algorithms.

forest and Cline forest methods to find out their effects on performance improvement. It was found that increasing the number of trees increases the success in both random forest and Cline forest. On the other hand, increasing the number of features increases performance only for the Cline forest and decreases the performance of the random forest.

Comparing Cline algorithm family to 22 single-tree algorithms and one multitree algorithm (random forest) based on eight data sets, the best four results were obtained with the Cline family. Bootsrapping procedure has positive effect on Cline forest's performances. Our experiments also include the training time analysis between algorithms. Cline forest is not the fastest algorithm but the best performing one.

In future work, we plan to improve performance by using different pruning methods, designing univariate Cline methods and using different methods when converting binary classifiers to N-ary classifiers.

The missing value problem will also be studied in the future. In practice, the missing values are filled by using statistical methods or the samples having missing value are ignored. Another approach is ignoring the corresponding feature from all samples. These approaches can be considered as preprocessing of the data. In decision forests, each tree has a different feature subset. In a tree, hyperplane features are usually the same in all the nodes. Trees using different features at each node can be created. During this process, the features having missing values can be ignored at a node. This would be different from handling missing features in a preprocessing stage.

ACKNOWLEDGMENT

The authors would like to thank B. Yanikoglu, B. Diri, S. Amasyali, and E. Islam Tatli for their help.

REFERENCES

[1] O. T. Yildiz and E. Alpaydin, "Model selection in omnivariate decision trees," in *Lecture Notes Artif. Intell.*, ser. 3720. Berlin, Germany: Springer-Verlag, 2005, pp. 473–484.

[2] X. B. Li, J. R. Sweigart, J. T. C. Teng, J. M. Donohue, L. A. Thombs, and S. M. Wang, "Multivariate decision trees using linear discriminants and tabu search," *IEEE Trans. Syst. Man Cybern. A, Syst. Humans*, vol. 33, no. 2, pp. 194–205, Mar. 2003.

[3] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Ann. Eugenics*, pp. 179–188, 1936.

[4] T. Kohonen, *Self Organizing Maps*, ser. Series in Information Sciences. New York: Springer-Verlag, 2001, p. 245.

[5] E. Alpaydin, *Introduction to Machine Learning*. Cambridge, MA: MIT Press, 2004, pp. 135–140.

[6] L. Breiman, "Random forests–random features," Dept. Statist., Univ. California, Berkeley, CA, Tech. Rep. 567, 1999.

[7] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*. New York: Chapman & Hall, 1984.

[8] T. S. Lim, W. Y. Loh, and Y. S. Shih, "A Comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms," *Mach. Learn.*, vol. 40, pp. 203–229, 2000.

[9] C. Blake and C. Merz, UCI Repository of Machine Learning Databases, [Online]. Available: http://www.ics.uci.edu/mlearn/ML-Repository.html 1998

[10] W. Y. Loh and Y. S. Shih, "Split selection methods for classification trees," *Statistica Sinica*, vol. 7, pp. 815–840, 1997.

[11] O. T. Yildiz and E. Alpaydin, "Linear discriminant trees," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 19, no. 3, pp. 323–353, 2005.

[12] M. F. Amasyali and O. Ersoy, "Cline: New multivariate decision tree construction heuristics," in *Proc. ICSC Congr. Comput. Intell. Methods Applicat.*, Istanbul, Turkey, Dec. 15–17, 2005, pp. 1–4.

[13] Y. Li, M. Dong, and R. Kothari, "Classifiability-based omnivariate decision trees," *IEEE Trans. Neural Netw.*, vol. 16, no. 6, pp. 1547–1560, Nov. 2005.

[14] C. T. Yildiz and E. Alpaydin, "Omnivariate decision trees," *IEEE Trans. Neural Netw.*, vol. 12, no. 6, pp. 1539–1546, Nov. 2001.

Evolved Feature Weighting for Random Subspace Classifier

Loris Nanni and Alessandra Lumini

***Abstract*—The problem addressed in this letter concerns the multiclassifier generation by a random subspace method (RSM). In the RSM, the classifiers are constructed in random subspaces of the data feature space. In this letter, we propose an evolved feature weighting approach: in each subspace, the features are multiplied by a weight factor for minimizing the error rate in the training set. An efficient method based on particle swarm optimization (PSO) is here proposed for finding a set of weights for each feature in each subspace. The performance improvement with respect to the state-of-the-art approaches is validated through experiments with several benchmark data sets.**

***Index Terms*—Ensemble generation, feature weighting, nearest neighbor, particle swarm optimization (PSO).**

I. INTRODUCTION

In several classification problems, in particular, when data are high dimensional and the number of training samples is small compared to the data dimensionality, it may be difficult to construct a good classification rule. In fact, a classifier constructed on small training sets has usually a poor performance.

In the literature, it is shown that a good approach to improve a weak classifier is to construct a strong multiclassifier that combines many weak classifiers with a powerful decision rule [10]–[12]. The most studied approaches for creating a multiclassifier are bagging, boosting, and random subspace. Bagging is based on a multiclassifier where each classifier is trained on a different subset of the training set and each subset is extracted randomly with replacement from the original training set [13]. Boosting is an iterative approach where each new classifier is focused on the training patterns that are misclassified by the previous classifiers [14]. Random subspace method (RSM) is based on selecting a random feature subset for each classifier [15].

In machine learning, feature weighting before classification is performed with the purpose to reveal the ability of each feature to distinguish pattern classes (see [7] for a survey). The weights of features are usually determined following two groups of approaches: through an iterative algorithm that evaluates, in a supervised manner, the performance of a classifier as a feedback to select a new set of weights [5], [9]; following a preexisting model's bias, e.g., conditional probabilities, class projection, and mutual information [6], [8].

In this letter, we investigate how particle swarm optimization (PSO) can be applied to generate an optimal set of feature weights for pattern classification to be coupled with a random subspace approach. PSO [1] is a relatively novel evolutionary computation technique based on the analysis of the social behavior of biological organisms, such as the movement of bird flocking [2].

The basic idea for applying PSO to random subspace is the following: for each feature in each subspace, we initially assign a random weight between 0 and 1, in this way, the dimension of each particle of the PSO problem is (number of features) $*$ (number of subspaces).

Manuscript received January 29, 2007; revised April 13, 2007 and June 28, 2007; accepted July 15, 2007.

The authors are with the DEIS, IEIIT—CNR, Università di Bologna, 40136 Bologna, Italy (e-mail: loris.nanni@unibo.it; lnanni@deis.unibo.it).

Color versions of one or more of the figures in this letter are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNN.2007.910737