Oblique decision trees using embedded support vector machines in classifier ensembles

Oblique Decision Trees Using Embedded Support Vector Machines in Classifier Ensembles

Vlado Menkovski, Ioannis T. Christou, and Sofoklis Efremidis

***Abstract*—Classifier ensembles have emerged in recent years as a promising research area for boosting pattern recognition systems' performance. We present a new base classifier that utilizes Oblique Decision Tree technology based on Support Vector Machines for the construction of oblique (non-axis parallel) tests on the nodes of the decision tree inducted. We describe a number of heuristic techniques for enhancing the tree construction process by better estimation of the gain obtained by an oblique split at any tree node. We then show how embedding the new classifier in an ensemble of classifiers using the classical Hedge(β) algorithm boosts performance of the system. Testing 10-fold cross validation on UCI machine learning repository data sets shows that the new hybrid classifiers outperforms on average by more than 2.1% both the WEKA implementation of C4.5 (J48) and the SMO implementation of SVM in WEKA. The application of the particular ensemble algorithm is an excellent fit for online-learning applications where one seeks to improve performance of self-healing dependable computing systems based on reconfiguration by gradually and adaptively learning what constitutes good system configurations.**

***Index Terms*—Classification, Decision Trees, On-line Learning, Supervised Learning**

I. Introduction

MANY different algorithms for induction of classifier models if trained with a big and diverse enough dataset perform with very high accuracy. But each has its own different strengths and weaknesses. Some perform better over discrete data, some with continuous, other classifiers have different tolerance for noise, and they have different speed of execution.

The work presented here is focused on improving known machine learning techniques by introducing new ways of combining the strengths of different approaches to achieve higher performance. We implemented an oblique tree classifier called SVMODT that exploits the benefits of multiple splits over single attributes and combines them with Support Vector Machine (SVM) techniques to take advantage of the accuracy of combined splitting on correlated numeric attributes.

To further boost the performance, particularly over noisy data and changing environments, we implemented a classifier ensemble with base classifiers, the SVMODT classifiers mentioned. The resulting system provides highly accurate classification in diverse and changing environments, as exhibited in extensive results with UCI Machine Learning repository datasets.

Manuscript received April 4, 2008.

Vlado Menkovski is with Athens Information Technology, 19.5 km. Markopoulou Ave., 19002 Peania, GREECE (e-mail: vmen@ait.edu.gr)

Ioannis T. Christou is with Athens Information Technology, 19.5 km. Markopoulou Ave., 19002 Peania, GREECE (phone: +30 210 6682725; fax: +30 210 6682703; e-mail: ichr@ait.edu.gr)

Sofoklis Efremidis is with INTRACOM S.A. Telecom Solutions, 19.7 km. Markopoulou Ave., 19002 Peania, GREECE (e-mail: sefr@intracom.gr)

II. Oblique Decision Trees

Tree induction algorithms like Id3 [3] and C4.5 [1] create decision trees that take into account only a single attribute at a time. For each node of the decision tree an attribute is selected from the feature space of the dataset which brings maximum information gain by splitting the data on its distinct values. As an example consider an attribute X_1 having values *a*, *b* and *c*. We split the data in three subsets where in the first X_1 is *a*, in the second *b* and in the third *c*. This attribute is selected as a best split for the current node of the tree if the split brings maximum information gain on the subset's classification. The information gain is calculated as the difference between the entropy of the initial dataset and the sum of the entropies of each of the subsets after the split.

Let info(T) be the average amount of information to identify a class of information in T. Then

- $\text{info}_x(T) = \sum_{i=1}^{n} \frac{|T_i|}{|T|} \cdot \text{info}(T_i)$ is the amount of information to identify each of the subsets
- gain(X) = info(T) – info_x(T) is the information gain by partitioning the data on test X

Id3 selects at each node the split on the attribute which gives the biggest gain.

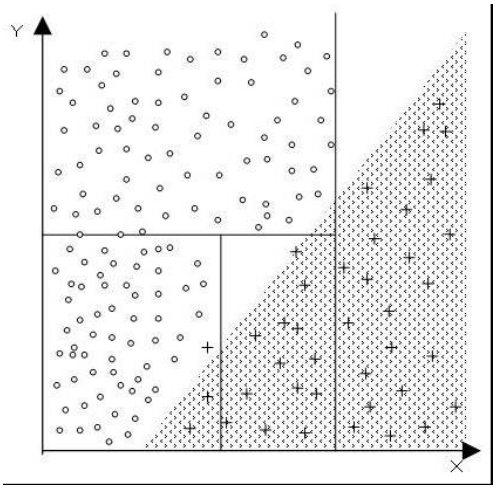

Fig. 1: Feature space of two attributes X and Y, several axis parallel splits and representation of the region of both classes.

Such trees make splits parallel to the axis in the feature space of the dataset. On the other hand, oblique decision trees split the feature space by considering combinations of the attribute values, be them linear or otherwise (see Fig.) [4]. Oblique decision trees have the potential to outperform regular decision trees because with a smaller number of splits an oblique hyperplane can achieve better separation of the instances of data that belong to different classes.

A. Using Support Vector Machines as Test Creators for Multiple Numeric Attributes

SVM builds an optimal hyper-plane (or multiple hyper-planes) separating the instances of data belonging to different sets, settings the hyper-plane's position so that it maximizes the margin of each class from the hyper plane, while minimizing the number of points lying on the "many" side of the hyper-plane. The result of the SVM technique is a hyper-plane described by the equation $\underline{w}^T\underline{x}+w_0 = 0$. This approach brings high accuracy in the separation of the data.

B. The ODT-SVM Algorithm

The algorithm that realizes the idea of building Oblique Decision Trees by using Support Vector Machines – ODT-SVM [4] is the basis for the J48-SVM-ODT implementation. The algorithm is given in Fig. 2 where X denotes the original input training data-set, N the cardinality of the input training data-set, and Nt the cardinality of the data-set of node t. The main thing to notice is that the embedded SVM algorithm is not applied to the whole attribute set of the training data-set available at the current node, but to the projection of this data-set on the sub-space of continuous attributes. Clearly then, the proposed algorithm is identical to classical C4.5 system on data-sets that consist of discrete attributes only.

```
Input:  a labeled training set
Output: an   Oblique   Decision   Tree   for
        Classifying New Instances

1. Begin with the root node t, having X(t) = X
2. For each new node t do
2.1. For   each   non-continuous   feature   xk
     k=1,…,l do
2.1.1. For each value akn of the feature xk do
2.1.1.1. Generate X(t)Yes and X(t)No according
         to the answer in the question: is
         xk(i)=akn,   i=1,2,…,Nt   (i   index
         running over all instances in X(t))
2.1.1.2. Compute the Impurity decrease
2.1.2. End-for
2.1.3. Choose the akn' leading to the maximum
       decrease with respect to xk
2.2.  End-for
2.3.  Compute the optimal SVM separating the
      points in X(t) into two sets X(t)1 and
      X(t)2 projected to the subspace spanned
      by all the continuous features xk,
      k=l+1,…,m
2.4.  Compute the impurity decrease associated
      with the split of X(t) into X(t)1, X(t)2,
      X(t)k
2.5.  Choose as test for node t, the test among
      2.1 – 2.4 leading to the highest impurity
      decrease
2.6.  If stop-splitting rule is met declare
      node t as leaf and designate it with a
      class label; else generate new descendant
      nodes according to the test chosen in
      step 2.5
3.End-for
4.Prune the tree
5.End
```

Fig. 2: ODT-SVM Algorithm.

C. Heuristics for Improving Performance

Many further improvements were introduced to boost the overall performance of the system. As with many classification algorithms decreasing the size of the decision tree brings better generalization; this is why most tree algorithms implement pruning heuristics. It is also beneficial to introduce pre-pruning techniques like stopping conditions in order to improve the performance of the induction if the further splits bring very little benefit to the classifier.

1) Pruning Principles

The partitioning method for tree induction will continue subdividing the set of training cases until each subset in the partition contains cases of a single class, or until no test offers any improvement. The result is often a complex tree that "over fits the data: by inferring more structure than is justified by the training cases" [1]. To overcome this problem and achieve better generalization a mechanism for pruning the decision tree is implemented. The pruning mechanism substitutes branches on an attribute with a leaf when the pruning criteria have been met. The pruning mechanism used is Reduced Error Pruning [7], which assesses the error rates of the tree and its components directly on the set of separate cases. For a given confidence level CF, the upper limit on the probability of error can be found from the confidence limits of the binomial

distribution; this upper limit is $U_{CF}(E, N)$ where N is the number of training cases and E is the number of training cases that are miss-classified. $N \times U_{CF}(E, N)$ is the predicted error for each leaf. If the sum of these errors is larger than the predicted error of the parent node the split is pruned and the parent node becomes the leaf. When a node is pruned it takes a sum of the distribution of all of its children and the class associated with this node is taken to be the class of the majority of all the training cases.

2) Clustering of Node Instances

The use of a hybrid of decision trees and linear combination splits like SVM gives possibilities of creating multiple hyper plane splits over the same feature space, achieving better precision. But the design of the SVM algorithms is such that the areas near the hyper plane have more errors than areas far away from the hyperplane. In other words we can classify data points with much higher accuracy when they are far from the separating hyperplane than when they are close to the hyperplane. The presence of these low certainty regions near the hyperplane lead to lower total accuracy of the SVM split. In order to overcome this effect the SVM-ODT algorithm clusters the training data of the current node using as single criterion the distance of the point from the separating hyperplanes found by SVM. Clustering is done via an application of the well-known very fast X-Means algorithm [9] which has the added advantage of automatically computing the optimal number of clusters in the problem set. The resulting clustering is taken to be a new split of the data set of the current node and its information gain is evaluated using the heuristics described above. In Fig. 3 the x-axis shows the distance of each point from the chosen SVM hyper-plane. Note that when more than 2 classes are involved, there are multiple hyper-planes produced, and clustering is done on a R^L vector space where L is the number of hyper-planes produced by the SVM algorithm.

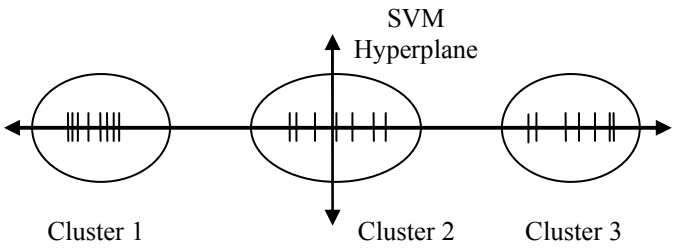

Fig. 3: SVM Clustering illustration.

The idea behind the clustering is to identify the areas of high accuracy against the areas of low accuracy. The low accuracy areas are expected to be near the hyperplane or spreading on both sides of the hyperplane. By splitting in this manner the high accuracy areas can end up as leaves in the SMV-ODT tree with high confidence on the classification and the low accuracy areas can be further spitted on a single or multiple attributes in order to divide it into more precise regions.

3) MDL-based Gain Computation

The Minimum Description Length (MDL) principle is a relatively recent method for inductive inference that provides a generic solution to the model selection problem. MDL is based on the idea that any regularity in the data can be used to compress the data, i.e., to describe it using fewer symbols than the number of symbols needed to describe the data literally. The more regularities there are, the more the data can be compressed [5].

MDL procedures automatically and inherently protect against over-fitting. The MDL principle is integrated into J48 such as that the value of particular information gain is lowered by the length of the message carrying the explanation of the split. This way complex splits are discouraged against simple splits of the data. The MDL principle in J48 is implemented by calculating the binary coding cost of the index of the attributed on which the split is proposed and the cost is subtracted from the information gain.

The SVM split in SVM-ODT algorithm uses more than one attribute for splitting so the MDL is implemented analogously to J48 by subtracting the coding of all the attributes used in the split from the information gain.

The SVM Clustering Split MDL is implemented by calculating the coding cost for the SVM Split as described plus adding to this the cost of coding the clusters. This value is subtracted from the information gain also.

The MDL principle should reduce the size of the trees and prevent over-fitting in the model but better compression of the SVM-ODT and SVM-ODT Clustering coding should bring less decrease in the information gain of these two algorithms and preferable results (this can be seen in the results section).

III. SVM-ODT Implementation Design

The implementation of the system uses the WEKA platform. WEKA (Waikato Environment for Knowledge Analysis) is a popular suite of machine learning algorithms written in Java and developed at the University of Waikato [2]. The WEKA platform provides a powerful set of utilities for loading of different types of datasets, architecture for classifiers, clusterers as well as functionalities for cross validation of the models. The platform also provides a GUI for executing the algorithm on different data sets as well as many other implementations of classification algorithms.

Besides providing an API for building and validating machine learning algorithms WEKA carries also implementation for many modern machine learning algorithms some of which are tightly integrated into the implementation of this system.

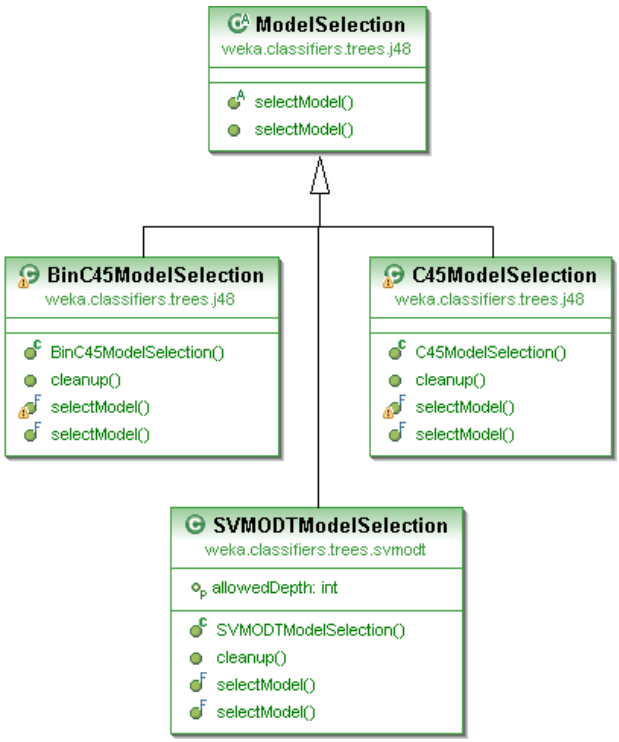

Fig. 5: UML Class diagram of Model Selection.

A. *J48-SVM-ODT*

J48-SVM–ODT is implemented as a class carrying the same name in the `weka.classifiers.trees` package. The class is slight modification of the J48 class which as mentioned before is the implementation of C4.5 in WEKA.

The modification consists of using `SVMODTModelSelection` [Fig. 4] for a model selector in the `buildClassifier` method instead of using `C45ModelSelection` as implemented in J48. The `SVMODTModelSelection` extents the `ModelSelection` class including other new types of splits SVM and the SVM clustering. The two added models for splitting are implemented in `SVMSplit` for SVM splits and `SVMClusteringSplit` for SVM clustering. Both implementations are extending `ClassifierSplitModel` [Fig. 5].

B. *SVM-ODT Split*

The SVM-ODT split is implemented in the `SVMSplit` class which takes all the continuous attributes from the feature set and builds a SVM using the `SMO` implementation in WEKA. Then the dataset is split into subsets one for each of the classes of the dataset. The information gain of the split is calculated and returned to the model selector to decide which split brings most gain.

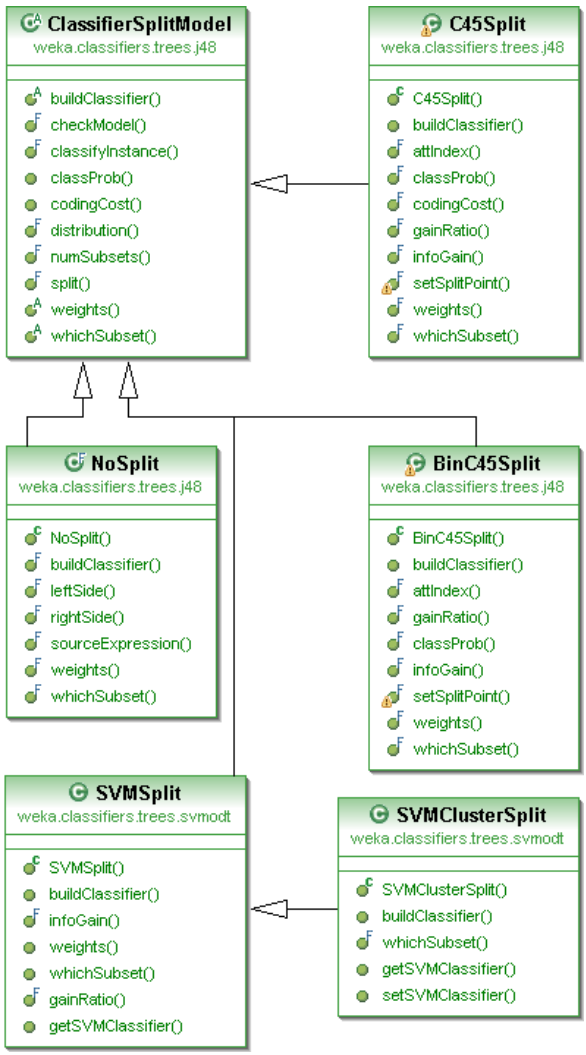

Fig. 6: UML Class diagram of Classifier Split Models.

C. *SVM-ODT ClusteringSplit*

The clustering model is implemented by the `SVMClusteringSplit` class. This class uses the previously generated SVM by the SVM-ODT split to improve performance avoiding recalculation the SVM hyperplane. Then a distance function calculates the distance from the (possibly more than one) hyperplane using the modified implementation of SMO called `ClusterableSMO` which is extended to include the `SVMDistance` method. The clustering algorithm used is `XMeans` the XMeans [9] clustering algorithm implementation for WEKA. `XMeans` returns the number of clusters as well as the cluster each data-point is assigned to. The result of the SVM-ODT clustering split is a number of subsets each one corresponding to a cluster of `XMeans`. The clusters far away from the hyperplane are expected to purely belong to one class and the clusters near or around the center ought to have mixed values.

IV. SVM-ODT Performance

The results from the testing of the J48-SVM-ODT algorithm with UCI M2 repository datasets are shown in TABLE I. All tables report accuracy obtained via 10-fold cross-validation.

TABLE I
RESULTS FROM TESTING THE J48-SVM-ODT ALGORITHM ON UCI DATASETS IN COMPARISON WITH SMO AND J48.

Dataset	J48-SVM-ODT	SMO	J48
anneal	98.44	97.44	98.44
anneal.ORIG	91.98	87.75	90.98
audiology	77.88	81.86	77.88
autos	81.95	71.22	81.95
balance-scale	95.52	87.68	76.64
breast-cancer	75.52	69.58	75.52
breast-w	96.14	97.00	94.56
colic	85.33	82.61	85.33
credit-a	86.38	84.93	86.09
credit-g	70.50	75.10	70.50
diabetes	76.43	77.34	73.83
glass	66.82	56.07	66.82
heart-c	80.20	84.16	77.56
heart-h	80.95	82.65	80.95
hepatitis	83.87	85.16	83.87
hypothyroid	99.58	93.61	99.58
ionosphere	92.02	88.60	91.45
Iris	95.33	96.00	96.00
kr-vs-kp	99.44	95.43	99.44
mushroom	100.00	100.00	100.00
segment	97.27	93.07	96.93
sick	98.81	93.85	98.81
sonar	72.12	75.96	71.15
splice	94.08	93.45	94.08
vehicle	74.82	74.35	72.46
vote	96.32	96.09	96.32
vowel	82.02	71.41	81.52
average	**87.03**	**84.90**	**85.88**

V. Using SVM-ODT in Online Learning Settings

In order to exploit the SVM-ODT algorithm in an Online Learning Settings we use it as a base classifier in an updatable classifier ensemble. The ensemble uses weight majority algorithm to boost the decisions of the more accurate classifiers. The weights of the base classifiers are continually updated based on their classification accuracy. As the environment changes the ensemble online adapts to the changes. Further improvements would be discarding individual classifiers and training new ones with the latest data points. The algorithm used for the classification ensemble is Hedge(β).

A. The Hedge(β)<SVMODT> System

Hedge(β) [6], [8] is a direct generalization of Littlestone and Warmuth's weighted majority algorithm [10]. The algorithm maintains a vector of weights for each classifier in its ensemble. The weights are values between 0 and 1 and they change overtime depending on the performance of the classifier the weight is assigned to.

This is an online learning algorithm which means that it adapts to the changes in the environment. The weights of the classifiers are scaled according to the current accuracy of the classifier. The classifier that is most accurate has the largest weight which can be interpreted as the probability of this classifier accuracy. When a new instance comes the classifiers all classify it, the classification distribution that Hedge(β) returns is a weighted sum of the distribution of all of the classifiers. When a feedback of the actual class of the instance is received the weights of the classifiers that misclassified the instance is lowered by the factor β. This way the algorithm continuously boosts the better classifiers.

An extension of the Hedge(β) algorithms discards classifiers that reached the low threshold of the weight and new classifiers are trained with only the most recent instances that will come in their place. This way the Hedge(β) algorithm benefits from the diversity of classifier assembles and also is capable of online learning to the changes in the environment.

This implementation for WEKA works with multiple classifiers of different types which are initially trained on slightly different datasets and weighted homogeneously.

VI. Ensemble Computational Results

The accuracy of the Hedge(β) algorithm on UCI datasets is given in TABLE II. For the simulation the ensemble consisted of ten J48-SVM-ODT instances. Each instance of a base classifier was trained on different 90% of the data and the accuracy reported was obtained using again 10-fold cross validation.

VII. Oblique Decision Trees Results Comparison

Other efforts have also explored use of Oblique Decision Trees for classification. Murthy, Kasif and Salzberg [11] describe the algorithm OC1 that uses deterministic hill climbing with randomization to find oblique hyperplane split and form nodes of an Oblique Decision Tree. The results of OC1 compared to J48-SVM-ODT are with lower accuracy as shown on couple of overlapping datasets (breast cancer, ionosphere, diabetes), only a slight difference on the Iris dataset (95.33 to 96%).

In the research of Setiono and Liu [12] a hybrid approach using Artificial Neural Nets and Decision Trees is showing improvement in performance over other tree classification algorithms. The results compared with J48-SVM-ODT on common datasets show that J48-SVM-ODT performs better. On the ionosphere dataset J48-SVM-ODT has 4% superior performance and on other datasets the results are close (breast cancer, iris, and diabetes).

An evolutionary approach is considered by Cantú-Paz and Kamath [13] for building Oblique Decision Trees. Even though the classification accuracy results do not over perform J48-SVM-ODT the stated speed of Decision Tree evolution can be very beneficial in specific implementations. Oblique

Decision Trees with nonlinear splits are explored by Ittner and Schlosser [14]. This interesting approach proves improvement in accuracy in particular datasets like the iris over J48-SVM-ODT but on average the performance is lower.

TABLE II
RESULTS FROM TESTING THE HEDGE(B) ALGORITHM ON UCI DATASETS.

Dataset	J48-SVM-ODT	SMO	J48	Hedge
anneal	98.44	97.44	98.44	98.78
anneal.ORIG	91.98	87.75	90.98	92.87
audiology	77.88	81.86	77.88	79.20
autos	81.95	71.22	81.95	86.83
balance-scale	95.52	87.68	76.64	96.16
breast-cancer	75.52	69.58	75.52	69.93
breast-w	96.14	97.00	94.56	97.14
colic	85.33	82.61	85.33	84.78
credit-a	86.38	84.93	86.09	85.22
credit-g	70.50	75.10	70.50	72.50
diabetes	76.43	77.34	73.83	76.17
glass	66.82	56.07	66.82	69.63
heart-c	80.20	84.16	77.56	80.86
heart-h	80.95	82.65	80.95	81.29
hepatitis	83.87	85.16	83.87	83.87
hypothyroid	99.58	93.61	99.58	99.58
ionosphere	92.02	88.60	91.45	92.59
iris	95.33	96.00	96.00	95.33
kr-vs-kp	99.44	95.43	99.44	99.37
mushroom	100.00	100.00	100.00	100.00
segment	97.27	93.07	96.93	97.75
sick	98.81	93.85	98.81	98.89
sonar	72.12	75.96	71.15	77.88
splice	94.08	93.45	94.08	94.20
vehicle	74.82	74.35	72.46	74.94
vote	96.32	96.09	96.32	96.55
vowel	82.02	71.41	81.52	85.96
average	**87.03**	**84.90**	**85.88**	**87.71**

VIII. CONCLUSIONS AND FUTURE DIRECTIONS

We have designed and implemented a classifier ensemble using Oblique Decision Trees combining C4.5 and Support Vector Machines that shows improvement over both C4.5 and SVM because it is capable of implementing several SVM splits and executing better separation in several regions. The addition of clustering SVM splits further improve the performance in cases where there is a region of uncertainty on the borders between separate regions.

Future improvements in the SVM-ODT classifications are also possible. The MDL implementation for the SVMSplit as well as the SVMClustering split brought some decreased performance in particular datasets. An effort to find a better compression for describing the split might result in improved performance as well.

IX. ACKNOWLEDGEMENTS

The work of this paper has been partially funded by the EU IST project DESEREC (IST-2004-026600).

REFERENCES

[1] Quinlan, J.R. C4.5 Programs for Machine Learning. Morgan Kaufmann, 1993.
[2] Weka Project home page, http://www.cs.waikato.ac.nz/ml/weka/.
[3] Quinlan, J. R. *Learning efficient classification procedures and their application to chess endgames*. In Machine Learning: An AI approach, R. Michalski, C. J., and T. Mitchell, Eds. Morgan Kaufmann., Los Altos, CA, 1983.
[4] Chistou I.T., Efremidis S. *An Evolving Oblique Decision Tree Ensemble Architecture for Continuous Learning Applications*. IFIP International Federation for Information Processing. Volume 247/2007, pp. 3-11. Springer Verlag, 2007.
[5] Rissanen, J. *A Universal Prior for Integers and Estimation by Minimum Description Length*. Journal Information for The Annals of Statistics. Vol. 11, No. 2, Jun., 1983.
[6] Kuncheva, Ludmila I. *Combining Pattern Classifiers: Methods and Algorithms*. John Wiley 2004.
[7] Quinlan, J.R. *Simplifying decision trees*. International Journal of Man-Machine Studies, 27(3):221–248, 1987.
[8] Schapire, Yoav Freund, and Robert E. *A decision-theoretic generalization of on-line learning and an application to boosting*. European Conference on Computational Learning Theory. 1995.
[9] Pelleg, D. Moore A. *X-means: Extending K-means with Efficient Estimation of the Number of Clusters*. In Proc. 17th International Conference on Machine Learning, 2000. 727-734.
[10] Warmuth, Littlestone, N., and Manfred K. *The Weighted Majority Algorithm*. IEEE Symposium on Foundations of Computer Science. *1*992.
[11] Murthy, Kasif, Salzberg. *A System for Induction of Oblique Decision Trees*. Journal of Artificial Intelligence Research 2 (1994) 1-32
[12] Rudy Setiono and Huan Liu. *A Connectionist Approach to Generation Oblique Decision Trees*. IEEE Transaction on Systems, Man, and Cybernetics – Part B: Cybernetics, Vol. 29, No. 3, June 1999
[13] Erick Cantú-Paz, Chandrika Kamath, *Inducing Oblique Decision Trees With Evolutionary Algorithms*. IEEE Transaction on Evolutionary Computation, Vol. 7, No. 1, February 2003.
[14] Andreas Ittner, Micheal Schlosser. *Non-Linear Decision Trees – NDT*. In: Proceedings of the 13th International Conferece on Machine Learning (ICML '96).