An Evolving Oblique Decision Tree Ensemble Architecture for Continuous Learning Applications

Ioannis T. Christou[1], and Sofoklis Efremidis[1]
1 Athens Information Technology
19 Markopoulou Ave P.O. Box 68 Paiania 19002 GREECE
{itc,sefr}@ait.edu.gr,
WWW home page: http://www.ait.edu.gr/faculty/I_Christou.asp

Abstract. We present a system architecture for evolving classifier ensembles of oblique decision trees for continuous or online learning applications. In continuous learning, the classification system classifies new instances for which after a short while the true class label becomes known and the system then receives this feedback control to improve its future predictions. We propose oblique decision trees as base classifiers using Support Vector Machines in order to compute the optimal separating hyper-plane for branching tests using subsets of the numerical attributes of the problem. The resulting decision trees maintain their diversity through the inherent instability of the decision tree induction process. We then describe an evolutionary process by which the population of base classifiers evolves during run-time to adapt to the newly seen instances. A latent set of base-classifiers is maintained as a secondary classifier pool, and an instance from the latent set replaces the currently active classifier whenever certain criteria are met. We discuss motivation behind this architecture, algorithmic details and future directions for this research.

1 Introduction

Classifier ensembles were first proposed a long time ago, but recently they have received a lot of attention in the machine learning community [1] because of their potential to overcome difficulties associated with any single algorithm's capabilities for a learning task. Classifier ensembles can be considered as meta-classifiers in that after the base classifiers reach their decisions, a final decision combining the various classifiers' results must be made. For this reason, the theoretical analysis of the power of classifier ensembles has been in general more difficult than that of individual learning algorithms. Nevertheless, classifier ensembles have been

successfully applied in many diverse areas ranging from multimedia and musical Information Retrieval [2] to Intrusion Detection [3] to recommender systems [4], etc.

One of the most important design decisions to be made in combining pattern classifiers is the choice of the base-classifier. In order to benefit the most from the combination of multiple classifiers, the ensemble should have sufficient diversity [1], for otherwise the decisions reached by the individual classifiers will be highly correlated and the probability that the performance of the overall system will be better than that of a single classifier will be slim. For this reason, unstable classifiers such as neural net-works and decision trees are often preferred as the base classification algorithms of a classifier ensemble.

In this paper we propose a classifier ensemble architecture suitable for online learning tasks in mixed-attribute domains where some attributes in the feature space are nominal whereas others are continuous-valued. We propose a modification of the classical C4.5 system architecture resulting in an oblique decision tree that branches on tests involving more than one continuous attribute using Support Vector Machines [5], and we present the details of the hybrid algorithm called SVM-ODT. We then propose new adaptive ensemble architecture for online learning applications using two evolving and alternating populations of SVM-ODT classifiers.

2 Building Oblique Decision Trees via Support Vector Machines

2.1 Decision Trees Overview

Tree classifiers work by constructing a decision tree for distinguishing data points between a finite set of classes. Starting from the root, decision tree construction proceeds by selecting an attribute from the feature space of the problem and splitting the data among two or more data sets depending on the values the selected attribute may take (Fig. 1).

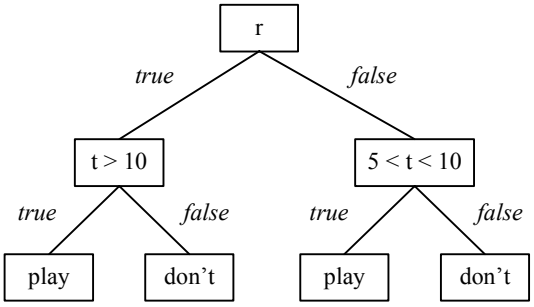

Fig. 1. A Decision Tree

Each "split" subset of the data set becomes a new node in the tree, under the current node. A node is considered a leaf node of the tree, and no further splitting occurs from this node on if the data set of this node is sufficiently "pure", for

example all data points in this data set belong to the same class. Selecting the attribute to split on can be done in many different ways reflecting different objective criteria that the method is supposed to optimize. The entropy-based measure of impurity dictates that the at-tribute to split on is the one providing maximum information gain: it is the attribute on which, if we split, the resulting children nodes maximize their "purity"; the split is locally optimal in dichotomizing the data set. The ID3 system, in particular [6], which is designed to work with nominal attributes only, works by producing splits on a node that completely use up the attribute selected. For example if an attribute can take on four different values in the data set, splitting on this attribute will result in four different children, one for each different value the attribute can take on. Since ID3 cannot work with continuous attributes, such attributes need to be discretized first. This discretization process can easily lead to low-performing classifiers. C4.5 extends the ID3 algorithm by allowing continuous variables to be split without completely consuming the attribute, and in most cases this results in serious performance gains.

Because decision trees have the ability to classify all data points in the training set with zero classification error, they belong to the category of "unstable" classifiers, meaning that small perturbations in the data set may lead to drastically different decision trees produced. This phenomenon is closely related to the generalization ability of a classifier to correctly classify previously unseen instances. Decision trees that obtain zero classification errors on the training set are more liable to overtraining, meaning the classifier has essentially "memorized" the training set instead of having "learned" concepts underlying the classification problem at hand. To improve the generalization capabilities of the classifier, pruning methods attempt to prune the decision tree after its initial construction so as to maintain small classification errors on the training set, but with expected enhanced accuracy of classification in new unseen instances. On the other hand, this inherent instability of decision trees makes them perfect candidates for the base classifiers of an ensemble classification scheme.

2.2 Oblique Decision Trees via Support Vector Machine

It is clear from the discussion above that in C4.5 and all standard decision tree classifiers the split of the feature space at each branching node occurs along axis-parallel hyper-planes, since every branching test involves only a single attribute (Fig. 2)

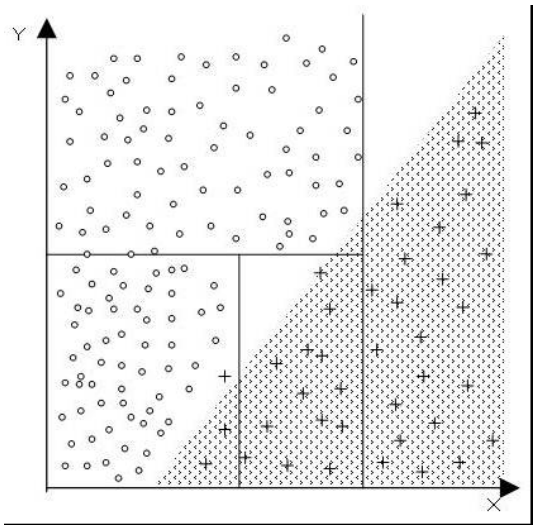

Fig. 2. Decision Trees split the data set of each node along axis-parallel hyper-planes

Quinlan [7] argues that for many application domains this restriction on the directions the splitting hyper-planes may take is not a problem, as evidenced by performance tests on a number of test-domains. Nevertheless, there are many application domains (especially ones where many of the domain attributes are continuous-valued) where decision trees are outperformed by more robust optimization methods such as Support Vector Machines (SVM).

SVM-based classifiers use rigorous mathematical programming theory to formulate the classification problem as an optimization problem in a vector space: the problem becomes that of finding the optimal separating hyper-plane that best separates the training set instances among the problem classes. SVM classifiers obviously work with data points belonging to $\mathbf{R}^n$. Problems involving attributes that are not continuous-valued must then be mapped somehow into a vector space and back. A popular technique for converting such problem sets into formats suitable for SVM optimization requires that each nominal attribute *attr* taking, say, m distinct values, be mapped into a set of m new {0,1} variables $e1...e_m$. A data point in the original space having for the nominal attribute *attr* the i-th discrete value, will be transformed into a point in an expanded vector space where the e_i variable for this point will take the value 1, and all the other e_j, $j \neq i$ variables for this attribute will be 0. There is a problem though with this technique. The data set corresponding to the problem for which the decision tree in Fig. 1 has been constructed is transformed in the vector space shown in Fig. 2. As can be seen, there is no single hyper-plane that will optimally decide which class a data point belongs to.

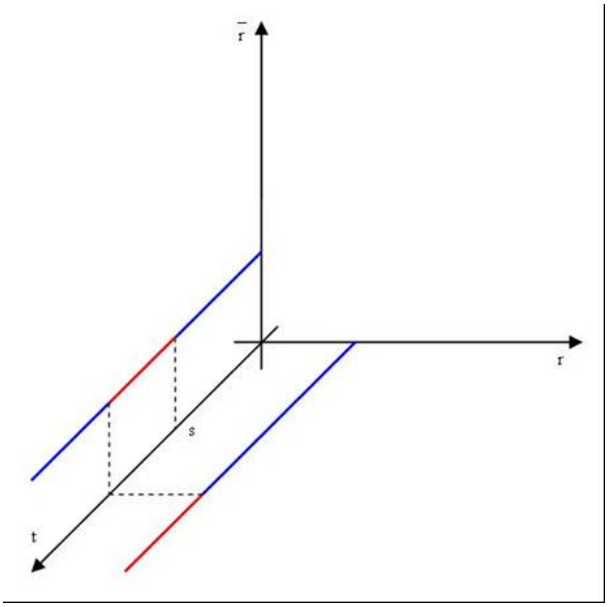

Fig. 3. Embedding a mixed-attribute space (t,r) into a higher (3) dimensional vector space

In general, trying to tweak a problem to fit into a domain that does not naturally fit in, is a practice that should be done with extreme care and only if there are no other tools available that can work directly with the problem domain.

Oblique Decision Trees are trees that branch using tests involving more than one at-tribute at any node. Many techniques for constructing oblique trees have been proposed in the past (see [8] for an approach building oblique trees using a random search procedure to combine attributes in each branching node). We propose to combine the strength of SVM in the continuous domain with that of decision trees in the discrete domain in an easy fusion.

In the following we discuss the 2-class classification problem, but it is easy to generalize the algorithm to deal with multiple classes. In particular, we propose an algorithm that is identical to ID3 except that its branching strategy is as follows: At every node, all the free continuous attributes of the problem are used to build a new problem set in the sub-space spanning all the continuous problem attributes. An SVM classifier is then built on the reduced continuous subspace, and an optimal hyper-plane separating the points of the current node's subset is constructed. The Information Gain of this split is then computed, along with the gains of the splits produced by the branching on each of the remaining non-continuous attributes. In the usual decision tree greedy manner, the split resulting in the highest gain is then selected and forms the test of the current node. The procedure is shown in detail in Fig. 4.

Algorithm ODT-SVM
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 x_k k=1,…,l do
2.1.1. For each value a_{kn} of the feature x_k do
2.1.1.1. Generate $X(t)_{Yes}$ and $X(t)_{No}$ according to the answer in the question: is $x_k(i)=a_{kn}$, i=1,2,…,N_t
2.1.1.2. Compute the Impurity decrease
2.1.2. End-for
2.1.3. Choose the $a_{kn'}$ leading to the maximum decrease with respect to x_k
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 *free (i.e. not currently constrained)* continuous features x_k, k=l+1,…,m
2.4. Compute the impurity decrease associated with the split of X(t) into $X(t)_1$ and $X(t)_2$
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 2 descendant nodes t1 and t2 according to the test chosen in step 2.5
3. End-for
4. End

Fig. 4. ODT-SVM algorithm

In fact, the algorithm is a template, defining a family of algorithms, in that different choices for measuring the impurity of a set or different stopping-splitting rules will lead to different algorithms. Moreover, step 2.3 can easily be modified so that instead of computing the optimal hyper-plane separating the data X(t) in node t projected in the subspace spanned by all the continuous features of the problem (thus likely consuming all continuous attributes in one test node) the test may select a subset of the set of continuous features –randomly or not– and compute the optimal SVM that separates the points of the node in that reduced subspace.

3 Evolving ODT-SVM Ensembles

Recently, classifier ensembles using pairs of classifiers trained of randomly chosen complementary sub-sets of the training set have been proposed in the literature as a means to improve both the stability as well as the diversity of the ensemble [9]. This approach leads to a pair of ensembles operating statically on the problem domain in that the population does not evolve after it has been trained on the training set. Similarly, evolving classifiers using Genetic Algorithms ideas has been proposed in [10] but the approach is not intended for online learning tasks.

For applications in online or continuous learning, we propose to use evolutionary methodology to evolve pairs of ODT-SVM ensembles, in the following way. A set S = $\{(c_1, c_1'), (c_2, c_2') \ldots (c_L, c_L')\}$ of L classifier pairs is first trained on the initially available training set T as follows: the first classifier of each pair i is trained on a randomly chosen subset of the training set $T_i \subseteq T$ and the second classifier of the pair is trained using the points $x_j \in T_i$ that were misclassified by the first classifier (the hard instances for the first classifier of the pair). The classifiers in each pair swap positions if the performance of the second classifier in the initial testing set is better than the performance of the first on the testing set.

During the online operation of the system (the continuous learning mode) new instances are given to the ensemble for classification. The system uses only the votes of the first classifier in each pair to reach a decision using any fusion or consensus strategy 1. However, all $2L$ classifiers classify the new instance, and when the true label for that instance becomes known (since the application is an online application) the classification accuracy of each classifier is updated to take into account the performance in the last received instance. When the performance of any of the top classifiers on a given number of the last arrived instances drops below the performance of its pair or some other criterion is met, the second and dormant classifier in the pair becomes the first active classifier and the original first is discarded. A new classifier is then trained on the instances the previously dormant classifier had missed and assumes the role of the dormant classifier of the pair.

The process is an evolving process with new classifiers being created and replacing old ones when those old classifiers' performance degrades. The system essentially remains static for as long as the ensemble's "knowledge" is adequate for the instances continuously arriving, but starts adapting itself to the new environment by modifying its population as soon as performance deteriorates enough. The process is depicted in Fig. 5. The decision maker could implement the Hedge(β) algorithm [1], as has been done successfully in [4]

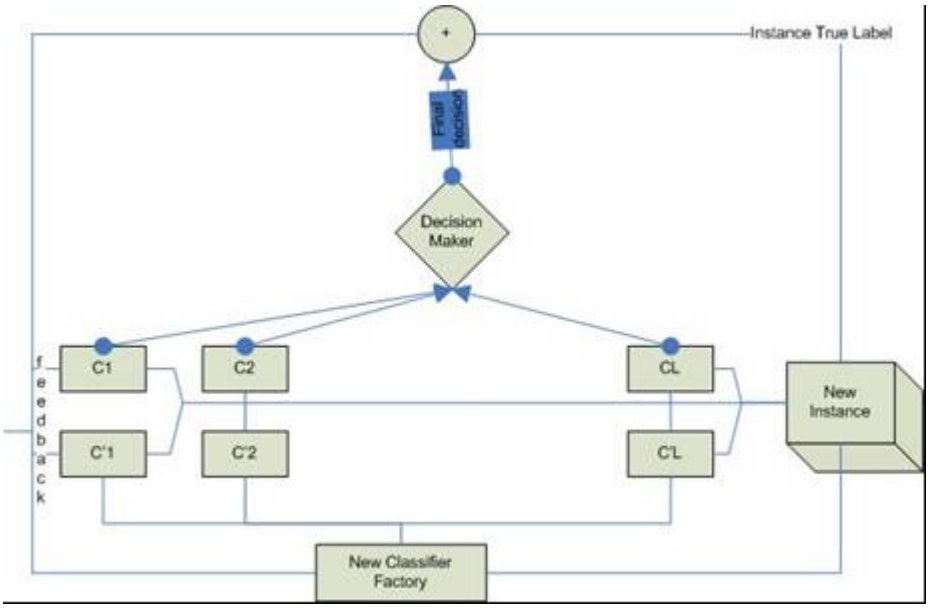

Fig. 5. Ensemble of ODT-SVM classifier pairs for online learning

4 Conclusions and Future Directions

We have presented an adaptive ensemble architecture for online learning tasks in changing environments. The architecture is based on Oblique Decision Trees using a modified C4.5 algorithm that treats the continuous attributes of a problem using Support Vector Machine technology while allowing for the discrete attributes of the same problem to be treated in the more natural decision tree philosophy. The Decision Tree philosophy of the base classifiers allows for more variety in the ensemble due to its inherent instability, variety which also comes from the fact that each base classifier in the ensemble is trained on a randomly selected subset of the training set.

For applications such as monitoring user profiles in the context of TV-program watching or movie-going recommendations, etc., the architecture has great promise in that it can follow the user's changing habits and adapt to them quickly enough so as to be very accurate most of the time. We plan to apply the system to the task of Anomaly Detection in surveillance systems using CCTV or other multi-media sources to reduce the number of false alarms while maintaining high accuracy rates.

References

1. Kuncheva, L. I. "Combining Pattern Classifiers – Methods and Algorithms", Wiley, Hoboken, NJ, 2004.
2. McKay C., et al: "ACE: A General Purpose Ensemble Classification Framework". Proceedings of the ICMC 05, 2005.

3. Koutsoutos S., Christou I.T. and Efremidis S.: "A Classifier Ensemble Approach to Intrusion Detection for Network-Initiated Attacks" - *invited contributed chapter in Emerging Artificial Intelligence Applications in Computer Engineering,* eds. John Soldatos et al, IOS Press, 2007.
4. Chistou I.T., Gkekas G., and Kyrikou, A.: "A Machine Learning Approach to the TV-Viewer Profile Adaptation Problem", submitted for publication, Dec. 2006.
5. Theodoridis, S. and Koutroumbas, K.: "Pattern Recognition", 3rd ed. Academic Press, San Diego, CA, 2006.
6. Quinlan, R.: "Induction on decision trees". Machine Learning, 1:1, 1986.
7. Quinlan, R.: "C4.5 Programs for Machine Learning". Morgan Kaufmann Publishers, San Francisco, CA, 1993.
8. Heath, D., Kasif, S., and Salzberg, S.: "Induction of Oblique Trees", IJCAI, 1993.
9. Kuncheva, L.I. and Rodriguez, J.J.: "Classifier Ensembles with a Random Linear Oracle", IEEE Transactions on Knowledge and Data Engineering, 19:4, 2007.
10. Ko, A. H.-R., Sabourin, R., and de Souza Britto, A. Jr.: "Evolving Ensemble of Classifiers in Random Subspace", Proc. Genetic and Evolutionary Computation Conf., GECCO 06, Seattle, WA, 2006.