Combination in Multi-View Forests using Dempster-Shafer Evidence Theory

Mohamed Farouk Abdel Hady, Friedhelm Schwenker and Günther Palm
Institute of Neural Information Processing
University of Ulm
James Franck Ring, D-89081 Ulm, Germany
Email: mohamed.abdel-hady, friedhelm.schwenker, guenther.palm@uni-ulm.de

***Abstract*—This paper considers ensembles of tree-structured classifiers using multi-view learning. An essential requirement to train an effective classifier ensemble is the diversity among the individual basic classifiers. In order to construct diverse individual classifiers, it is assumed that the object to be classified is represented through multiple feature sets (views). Different views of the data can then be combined to improve the accuracy of the learning task. For the fusion of the individual decisions (of tree classifiers in this case) the Dempster-Shafer approach has been investigated. Numerical experiments have been performed on two benchmark data sets: a data set of handwritten digits, and another data set of 3D visual objects. Results show that multi-view learning can improve the performance of the individual tree classifiers.**

Keywords: Dempster-Shafer Evidence theory, pattern recognition, multi-view learning, radial basis function neural networks, multi-class decomposition.

I. Introduction

The growing interest in combining classifiers is due to the fact that computing a best individual classifier for a classification task is difficult from a computational and statistical perspective. In addition, the use of a multiple classifiers allows the exploitation of complementary discriminating information that the group of classifiers may provide. Combining such a group of classifiers attempts to produce a more accurate classifier decision than a single classifier of the group. Combining classifiers to achieve higher accuracy is an important research area in a number of communities such as machine learning, pattern recognition, and artificial neural networks, and appears under different notions in the literature, e.g. multiple classifier systems (MCS), classifier fusion, classifier ensembles, divide-and-conquer classifiers, mixture of experts [1]. Error diversity is an essential requirement to build an effective classifier ensemble. Diversity among classifiers means that they have independent (uncorrelated) errors. There is no unique diversity measure, e.g. in [2] ten different measures have been proposed by Kuncheva, such as correlation coefficient, disagreement measure and doubt-fault measure. Obviously, there is a trade-off between the diversity of the classifiers and their accuracies. Many techniques for constructing ensembles consisting of diverse individual classifiers have been developed. One technique is based on combining classifiers trained on different training sets, i.e. bagging [3] and boosting [4]. Another technique to promote the diversity is based on combining classifiers trained on different feature subsets, such as Random Forests [5]. Feature subset selection is time-consuming and sometimes deleting parts of the feature set degrades the performance of the individual classifiers. Hence, this technique is efficient only if the features are redundant. In this paper, a new ensemble method, denoted as *Multi-view Forests*, is proposed, a set of tree-structured classifiers are chosen as the base classifiers. Tree classifiers can be used to decompose multi-class recognition problems into less complex binary classification subtasks. Dempster-Shafer evidence theory can be applied as soft combination rule to fuse the intermediate results of the node classifiers and to produce the final decision of each tree. After that it can be used again to combine the results of the different trees in the forest. The rest of the paper is organized as follows: In Section II and III Multi-view Forests and the evidence-theoretic approach are explained, and in Section IV results of applying the proposed method on object recognition are presented. Section V offers the conclusion.

II. Multi-View Forests

A *multi-view forest* is an ensemble of tree-structured classifiers, and a tree classifier can be seen as a hierarchical ensemble of binary classifiers which solves a multi-class classification task using a single feature set (called *Single-View Tree*) or even a group of feature sets, then it is called *Multi-View Tree*. Let $L = \{(x_{\mu 1}, \ldots, x_{\mu n}, y_\mu) | x_{\mu v} \in X_v, v = 1, \ldots, n$ and $y_\mu \in \Omega, \mu = 1, \ldots, N\}$ be the training data set $(x_\mu, y_\mu) : \mu = 1, \ldots, N$ where x_μ is a data point described by n feature sets, where y_μ denotes the class label of x_μ and $\Omega = \{\omega_1, \ldots, \omega_K\}$ is the predefined set of classes.

A. Multi-View Learning

Multi-view learning is a machine learning approach where each pattern is represented by many features obtained through different physical sources and sensors or derived by different feature extraction procedures. For example, a web page can be represented by different views, e.g. a distribution of words used in the web page, hyperlinks that point to this page, and any other statistical information. Multi-view learning was introduced for semi-supervised learning by Blum and Mitchell for Co-training [6], where they proved that having multiple representations of a particular object can improve the classifier

performance using unlabeled data. Multi-view learning has been applied in clustering as well, for instance Gupta and Dasgupta [7] proposed a multi-view hierarchical clustering algorithm (see Algorithm 2 and Figure 1). It depends on the assumption that different views may have different distance measures leading to different clusterings. This method seems to work better than taking a linear combination of the distance measures, or appending the different feature sets together. In [7], the tree structure has been constructed through a top-down approach using the best feature set at each split point in the tree. To select the best feature set, agglomerative clustering is applied to each feature set to produce a pair of clusters. Intuitively, the required best feature set is the one that provides the most well-separated clusters. A scale and feature independent criterion, *evaluate*, to measure the separation quality of a pair of clusters is defined by the ratio of the inter-cluster distance to the sum of inner-cluster distances.

B. Tree-Structured Multi-class Decomposition

Many real-world problems are multi-class problems, e.g. optical character recognition. Typically these multi-class problems are decomposed into multiple two-class classification problems. In the *One-against-Others* decomposition scheme, a K-class problem is decomposed into K two-class problems where a binary classifier is trained for each class to discriminate it from the other K-1 classes. A major drawback of this approach is the *False Positives*. During the classification phase, it is expected that exactly one of the K classifiers replies a positive answer but for large K often more than one classifier reply positively which is a tie that must be broken by additional criteria. The *One-against-One* scheme is another straightforward approach to solve the multi-class problem. Here for each pair of classes a binary classifier is trained, and therefore, to solve a K-class classification problem, $K(K$-1)/2 binary classifiers have to be trained and evaluated in the classification phase. If K is large, the number of classifiers is very large compared to the *One-against-Others* scheme.

The task of the tree-structured approach is to decompose a given K-class problem into a set of simpler K-1 binary problems and to train classifiers to solve the binary problems at the internal nodes within the tree through a base learning algorithm ($BaseLearn$). In the classification phase, for a given instance x, the intermediate results of the internal classifiers are combined through a given combination method ($TreeCombiner$) to produce the final decision of the ensemble.

The approach works as follows: First, the set of K classes (Ω) is split into two disjoint subsets, known as meta-classes or super-classes. Then these meta-classes are again split recursively until each meta-class contains one of the original classes. The resultant binary tree has K leaf nodes, one for each original class and K-1 internal nodes, each associated with two meta-classes and a binary classifier. (See Algorithm 1).

1) Tree Training Phase: In the first step, the tree is generated by applying a splitting method recursively to split the set of classes at each node of the tree into two disjoint subsets, until every subset contains exactly one class. The set of classes can be split randomly into two disjoint subsets but often some classes are similar (called confusion classes). For example, in optical character recognition, *o*, *O* and *Q* might be considered as members of a certain confusion class due to their visual similarity. Therefore, it is recommended

Algorithm 1 Tree Ensemble Learning Algorithm

Require: L - set of m labeled training examples
$\Omega = \{\omega_1, \ldots, \omega_K\}$- set of classes
$BaseLearn$ - base learning algorithm
$TreeCombiner$ - hierarchical combination method
Training Phase
$\Omega_1 \leftarrow \Omega$
Generate Class Hierarchy as follows:
1) $C \leftarrow \{(c_k, \omega_k)\}_{k=1}^{K} \leftarrow GetClassCentroids(L)$
2) $hierarchy \leftarrow BuildNode(\Omega_1, C)$
for each internal node j at $hierarchy$ **do**
Filter the training set L as follows:
$L_j = \{(x, y) | x \in \Omega_j$ and $y = 0$ if $x \in \Omega_{2j}$ and $y = 1$ if $x \in \Omega_{2j+1}\}$
Train binary classifier, $h_j \leftarrow BaseLearn(L_j)$
end for
Prediction Phase
return $TreeCombiner(x, hierarchy)$ for a given instance x

Algorithm 2 $BuildNode$ - (Bottom-Up Approach)

Require: Ω_j - set of classes assigned to tree node j
C_j - set of centroids of classes in metaclass Ω_j
if $|\Omega_j|$ = 1 **then**
Add a leaf node j to $hierarchy$ that represents class Ω_j
else
Add an internal node j to $hierarchy$ that represents metaclass Ω_j
for $v = 1$ to n **do**
Initially, put each class in Ω_j in a separate cluster
repeat
Get the two most close clusters in Ω_j
Merge these two clusters into a new cluster
until the number of remaining clusters is two
Denote the remaining clusters, $\Omega_{2j}^{(v)}$ and $\Omega_{2j+1}^{(v)}$
$score_v \leftarrow evaluate(\Omega_{2j}^{(v)}, \Omega_{2j+1}^{(v)})$
end for
Get the winner view, w = arg $max_{1 \le v \le n} score_v$
$C_{2j} \leftarrow$ set of centroids of classes in $\Omega_{2j}^{(w)}$
$BuildNode(\Omega_{2j}^{(w)}, C_{2j})$
$C_{2j+1} \leftarrow$ set of centroids of classes in $\Omega_{2j+1}^{(w)}$
$BuildNode(\Omega_{2j+1}^{(w)}, C_{2j+1})$
end if
return $hierarchy$

Algorithm 3 $BuildNode$ - (Top-Down Approach)

Require: Ω_j - set of classes assigned to tree node j
C_j - set of centroids of classes in metaclass Ω_j

if $|\Omega_j| = 1$ **then**
  Add a leaf node j to $hierarchy$ that represents class Ω_j
else
  create an internal node j that represents metaclass Ω_j
  Add $node_j$ to $hierarchy$
  for $v = 1$ to n **do**
    Get the two most distant classes in Ω_j: $(c_{j1}, \omega_{j1}), (c_{j2}, \omega_{j2})$
    $\{\Omega_{2j}^{(v)}, \Omega_{2j+1}^{(v)}\} = seeded\text{-}k\text{-}means(C_j, c_{j1}, c_{j2})$
    $score_v \leftarrow evaluate(\Omega_{2j}^{(v)}, \Omega_{2j+1}^{(v)})$
  end for
  Get the winner view, w = arg $max_{1 \leq v \leq n} score_v$
  $C_{2j} \leftarrow$ set of centroids of classes in $\Omega_{2j}^{(w)}$
  $BuildNode(\Omega_{2j}^{(w)}, C_{2j})$
  $C_{2j+1} \leftarrow$ set of centroids of classes in $\Omega_{2j+1}^{(w)}$
  $BuildNode(\Omega_{2j+1}^{(w)}, C_{2j+1})$
end if
return $hierarchy$

practically to do the splitting using some similarity measure or by applying a clustering method in order to keep similar classes together in the same subset. There is a number of various ways to measure the similarity between two classes such as Nearest Neighbor (Single linkage), Farthest Neighbor, Average Distance and Centroid. In this study, the distance between classes ω_i and ω_k is the Euclidean distance between the centroid of the training examples that belong to class ω_i and that of the training examples belonging to class ω_k.

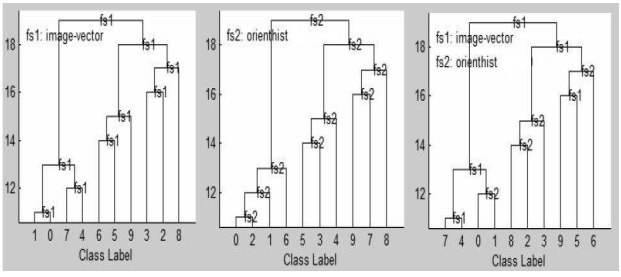

Figure 1. Dendrogram constructed for digits data using: *image-vector* (fs_1), and orientation histograms (fs_2).

In addition to hierarchical clustering, k-means clustering is used to split the set of classes into disjoint subsets to generate a hierarchy of RBF networks (see Algorithm 3). The different splits are evaluated using a splitting evaluation measure in an attempt to find the best view to split the set of classes. Impurity measures such as the Entropy or Gini index or the ratio of inter-class distance and intra-class distance are examples for such criteria. In this paper, both multi-view hierarchical clustering (bottom-up) and k-means clustering (top-down) are used to generate the tree structure. In the second step of the tree training phase, the binary classifiers which were assigned to the internal nodes have to be trained by using supervised learning procedures. As binary classifiers, support vector machines, artificial neural networks such as radial basis function (RBF) neural networks can be used (Fig. 2).

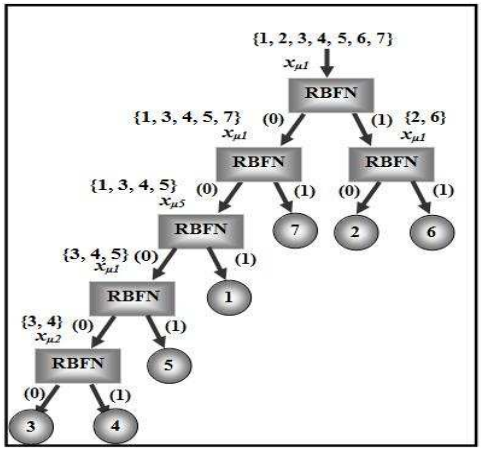

Figure 2. *Multi-View Tree* constructed for Fruits data using: Color histograms ($x_{\mu 1}$), orientation histograms utilizing canny edge detector ($x_{\mu 2}$) and orientation histograms using opponent colors red and green ($x_{\mu 5}$).

III. Tree Classification Phase

Two different strategies to combine the decisions of tree-structured binary classifiers have been used throughout this study: decision-tree-like combination and a combination scheme which is derived from the Dempster-Shafer rule of combination. These schemes are rather different in terms of complexity and output type (crisp vs. soft). A simple and fast method to get the final result from the tree is the decision tree approach where the tree is traversed following the individual node classifiers starting from the root node to a leaf node which is then representing the classification result. Each classifier decides which of its child node has to be evaluated next, until a leaf node is reached. This combination method is very fast. Drawbacks of this approach are: (1) Misclassification of high-level classifiers can not be corrected; (2) it does not benefit from the discriminating information, provided by the classifiers, outside the path; (3) the output is a class label, it does not provide fuzzy class memberships, and (4) majority voting is the only possible classifier combination technique.

A. Dempster-Shafer evidence theory

It is a mathematical theory for representing and combining evidences, which was introduced by Dempster [8] and Shafer [9]. The reasons for using results derived from this theory in the multiple classifiers combination are the ability to discriminate between ignorance and uncertainty, the ability to easily represent evidences at different levels of abstraction and the possibility to combine evidences from different sources. The Dempster-Shafer theory starts by assuming a universe of discourse called the *frame of discernment* that consists

of a finite set of K mutually exclusive atomic hypotheses $\theta = \{\theta_1, \ldots, \theta_K\}$.

Let 2^θ denote the set of all subsets of θ. Then a *mass function* over the frame of discernment θ is a function $m : 2^\theta \rightarrow [0, 1]$ that is called *basic probability assignment* (*bpa*) if it satisfies the following conditions:

$$m(\phi) = 0 \quad \text{and} \quad \sum_{A \subseteq \theta} m(A) = 1 \tag{1}$$

Dempster's rule of combination combines the basic probability assignments produced by n independent sources $m_1, \ldots, m_n$ using the orthogonal sum.

$$m(A) = \sum_{\cap A_i = A} \prod_{1 \leq i \leq n} m_i(A_i) \tag{2}$$

In [10] it was applied to prototype-based classifiers and calculates the belief functions from the distance measures of different classifiers which are then combined utilising Dempster-Shafer evidence theory. As distance measures the inter-class-distances and intra-class-distances were used. Another approach similar to decision templates is used in [11] to calculate the degree of belief. The distances between the classifier outputs for the sample to be classified and the mean classifier outputs calculated on the training samples are transformed into basic probability assignments. The so calculated evidences are then combined using the orthogonal sum.

In order to combine an ensemble of tree-structured binary classifiers using the Dempster-Shafer theory, it is assumed that θ_k = "the hypothesis that a given instance belongs to class ω_k" and that the internal node classifiers are the sources of evidence. First, basic probability assignments m_j must be derived from the outputs of the individual classifiers h_j within the tree. Usually, not all classifiers produce outputs that satisfy the conditions of probability assignments. In these cases the outputs are transformed into basic probability assignments (bpas) as follows: (1) all negative values are set to zero, (2) if the sum of a classifier outputs is greater than one, it is normalised to sum up to one. In order to account for ignorance, the difference of the sum of the output values to one is assigned to Ω ($m_j(\Omega)$). Then the resulting bpas m_j of all classifiers are combined using *Dempster's rule of combination* without normalization.

B. Discounting Technique

Discounting is used to propagate the outputs of high-level classifiers to the classifiers at the lower levels. That is, the output of each internal node classifier is multiplied by the discounted output of the respective predecessor classifier where the root node classifier output is not discounted. The motivation for discounting is the fact that a number of classifiers will be enforced to respond to instances that actually belong to classes that are unknown to them. For example, a classifier f_j that discriminates between $\Omega_{2j} = \{1, 5\}$ and $\Omega_{2j+1} = \{2, 6\}$ has to label an instance x belonging to class 3. In this case, it is desirable that $f_{j1}(x)$ and $f_{j2}(x)$ tends to zero but at the real situation, $f_{j1}(x)$ or $f_{j2}(x)$ may tend to one. If at least one classifier within a certain path gives a low response to instance x, this leads to weaken any undesirable high responses. In contrast to the decision-tree-like approach, all the classifier within the tree, are participating in classifying an input sample and the final output estimates a fuzzy-like class membership of the input to all classes (soft class label).

IV. Numerical Evaluation

A. Results on the Fruits Data Set

The Fruits data set consisting of seven different objects with 120 images per class was used for classifier evaluation. Five different feature sets were extracted: color histograms (fs1), orientation histograms utilising canny edge detection (fs2), utilising sobel edge detection (fs3), utilising opponent colors black and white (fs4) and utilising opponent colors red and green (fs5) (see [12] for more details). Experiments were performed by 10 runs of 10-fold cross-validation (CV). First, the views have to be selected from the set of all possible views. For instance, random View Selection can be used. Another option is to construct a tree classifier for each possible view, leading to 31 classifiers for 5 feature sets. For the representation of feature sets, the binary string representation is chosen. In this representation, each subset is represented by N bits (N: number of features in the full set). Each bit represents presence (1) or absence (0) of that feature set in the subset. For example, if N=4, then string 1001 will represent subset $\{$fs1, fs4$\}$. RBF

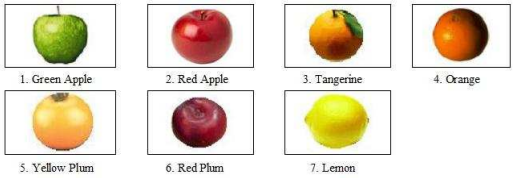

Figure 3. A sample of the images in the fruits data set

networks with 16 RBF neurons were used as binary classifiers. The Bottom-Up Approach defined in Algorithm 2 is used to build the class hierarchies. The initial centers of the RBF neurons are determined by performing class-specific k-means clustering, the widths of the RBF neurons are determined by estimating the variance of the data points belonging to the respective clusters. The output weights are learned by calculating the pseudo-inverse solution of the least squares approach that is a fast method yielding good classification results. The decision-tree-like and Dempster-Shafer approach were applied together with majority vote, min, max, sum and product rule. Table I shows the results of using all the possible views in building tree classifiers. The first column shows the rank of the tree classifier of the sorted list. The second column contains the views used by the tree classifier, represented as a binary string indicating whether a view is in use or not. The third and the fourth column list the CV test set accuracy of each tree classifier for decision-tree-like (DT) and Dempster-Shafer-based (DS) combination, respectively.

Table I
MEAN ACCURACY AND STANDARD DEVIATION OF THE TREE CLASSIFIERS

Rank	View	accuracy	
		DT	DS
1	[1 1 0 0 1]	96.8 ± 1.66	97.2 ± 1.55
2	[1 1 1 0 1]	96.5 ± 1.82	96.9 ± 1.72
3	[1 1 0 0 0]	96.5 ± 1.92	96.9 ± 1.71
4	[1 0 1 0 1]	96.4 ± 1.82	96.8 ± 1.77
5	[1 0 0 1 0]	96.4 ± 1.96	96.4 ± 2.02
6	[1 0 1 1 0]	96.4 ± 1.96	96.4 ± 2.02
7	[1 1 0 1 1]	96.4 ± 1.99	96.4 ± 2.08
8	[1 1 1 1 1]	96.4 ± 1.99	96.4 ± 2.08
9	[1 0 0 1 1]	96.4 ± 1.98	96.4 ± 2.07
10	[1 0 1 1 1]	96.4 ± 1.98	96.4 ± 2.07
11	**[1 0 0 0 0]**	**96.4 ± 2.03**	**96.6 ± 1.86**
12	[1 1 0 1 0]	96.4 ± 1.97	96.4 ± 2.03
13	[1 1 1 1 0]	96.4 ± 1.97	96.4 ± 2.03
14	[1 1 1 0 0]	96.2 ± 2.04	96.7 ± 1.82
15	[1 0 1 0 0]	96.1 ± 1.99	96.6 ± 1.80
16	[0 1 1 1 1]	96.1 ± 2.29	96.2 ± 2.30
17	[0 0 1 1 1]	96.1 ± 2.31	96.2 ± 2.32
18	[1 0 0 0 1]	96.0 ± 1.94	96.4 ± 1.80
19	[0 1 0 1 1]	95.3 ± 2.61	95.2 ± 2.66
20	[0 0 1 0 1]	95.2 ± 2.36	95.6 ± 2.31
21	[0 0 0 1 1]	95.2 ± 2.67	95.1 ± 2.69
22	[0 1 0 0 1]	94.6 ± 2.23	95.1 ± 2.20
23	[0 1 1 0 1]	94.3 ± 2.55	94.7 ± 2.48
24	[0 1 0 1 0]	94.2 ± 2.52	95.2 ± 2.51
25	[0 1 1 1 0]	94.1 ± 2.57	95.2 ± 2.55
26	[0 1 1 0 0]	92.2 ± 2.99	92.8 ± 3.05
27	**[0 1 0 0 0]**	**91.6 ± 2.70**	**92.2 ± 2.74**
28	[0 0 1 1 0]	90.2 ± 3.26	90.7 ± 3.00
29	**[0 0 1 0 0]**	**89.7 ± 4.08**	**90.3 ± 4.12**
30	**[0 0 0 0 1]**	**89.7 ± 3.24**	**89.8 ± 3.28**
31	**[0 0 0 1 0]**	**88.5 ± 3.38**	**88.8 ± 3.29**

Table II
MEAN AND STANDARD DEVIATION OF CV TEST SET ACCURACY OF MULTI-VIEW FOREST CONSISTING OF THE FIVE SINGLE-VIEW TREES (IN BOLD IN TAB. 1)

TCM	FCM	MVF_{single}
DT	MV	98.6± 1.35
DS	MV	98.8 ±1.29
	Min	98.6 ± 1.46
	Max	99.1 ± 0.98
	Mean	99.2 ± 0.89
	Prod	99.1 ± 1.15
Best Tree		96.6± 1.86
Gain		2.58%

Table II illustrates the classification results of the ensembles constructed by combining the five single-view tree classifiers (MVF_{single}). The ensembles combine the outputs of tree classifiers (produced by DT and DS respectively) using Majority Voting (MV), minimum (Min), maximum (Max), mean (Mean) and product (Prod) rules as forest combination methods (FCM), respectively.

Table IV-A illustrates the classification results of the ensemble constructed using the first, the middle and the last 10 tree classifiers in the sorted list, respectively (MVF_1, MVF_2, MVF_3). First, the accuracies of the five Single-View Trees and an ensemble of them are compared. From Table I, it can be seen that the tree classifier based only on fs_1 lies at rank 11, tree classifier based only on fs_2 lies at rank 27, tree classifier based only on fs_3 lies at rank 29, tree classifier based only on fs_5 lies at rank 30 and finally comes tree classifier based only on fs_4 lies at rank 31. This means that the tree classifier based on fs_1 outperforms all other single-feature-set classifiers by about 4.5%. From Table II, the best ensemble has an accuracy of 99.2% ± 0.89. Therefore, the ensemble of the Single-View Trees outperforms the best single individual classifier. The reason of this performance is the large diversity between the classifiers as each of them use different feature set. Second, the results of the Single-View Trees and the Multi-View Trees are compared. From Table I, we can observe that the best Single-View tree classifier, based only on feature (fs_1), is at rank 11, thus 10 Multi-View tree classifiers outperform the best single-view classifier. The tree classifier based on feature fs_1, fs_2, fs_5, is at first rank, and achieves an accuracy of 96.8% ± 1.66 (DT) or 97.2% ± 1.55 (DS). So it outperforms the corresponding single view tree classifiers.

Table III
MEAN AND STANDARD DEVIATION OF THE MULTI-VIEW FORESTS

TCM	FCM	MVF_1	MVF_2	MVF_3
DT	MV	96.4 ±1.98	97.2±1.66	97.9± 1.59
DS	MV	96.6± 1.94	97.3± 1.62	98.0± 1.55
	Min	97.8± 1.39	98.8± 1.19	97.9± 1.70
	Max	97.8± 1.45	98.9± 1.02	98.1± 1.39
	Mean	97.7± 1.45	98.7± 1.10	98.7± 1.25
	Prod	97.8± 1.45	98.8± 1.00	98.7± 1.31
Best Tree		97.2± 1.55	96.6± 1.86	95.2± 2.67
Gain		0.68%	2.27%	3.53%

Third, we compare between Multi-View Trees and Ensemble of Multi-View Trees. From Table , we can see that the best ensemble, based on the 10 most accurate tree classifiers, achieves an accuracy of 97.8% ± 1.39 (DS + Min) while the best of the 10 trees has a rate 97.2% ± 1.55(DS). Therefore, there is a gain in accuracy only 0.68%. For the second ensemble, based on the second 10 tree classifiers in the list, the best result is 98.9% ± 1.02 (DS + Max). This means that the gain in accuracy is 2.2%. For the last ensemble, based on the following 10 tree classifiers in the list (weaker classifiers), the best rate is 98.7% ± 1.25 (DS + Mean) with a gain about 3.5%.

Finally, we compare among the three constructed ensembles. From Table I, we can find that the ten classifiers of ensemble MVF_1 use fs1 as best feature set in about 4 of their 6 binary classifiers while only 6 trees of the ten of MVF_2 use fs1. Therefore, ensemble MVF_2 is more diverse than MVF_1 as it contains weaker and less identical tree classifiers. For this reason, ensemble MVF_2 has more gain than MVF_1 and ensemble MVF_3 gains more than MVF_2. The weaker the combined individual classifiers, the higher the gain of the ensemble accuracy will be. Although the ensemble MVF_3 is consisting of less accurate individual classifiers than that of MVF_1 and MVF_2, the observed gain of MVF_3 is higher than that of MVF_1 and MVF_2 and also the classification accuracy of MVF_3 outperforms the other two ensembles, in many cases.

B. Results on the Handwritten Digits

The performance was evaluated using the handwritten STATLOG digits data set [13]. This data set consists of 10,000 images (1,000 images per class) and each image is represented by a 16x16 matrix containing the 8-bit gray values of each

Table IV
DESCRIPTION OF THE VIEWS OF THE HANDWRITTEN DIGITS

Feature	Description
image-vector	A 256-dim vector results from reshaping the 16x16 image matrix.
orienthisto	A 144-dim vector that represents 9 orientation histograms where an image matrix has been divided into 3x3 overlapped sub-images and a histogram is created for each sub-image.
pca-40	A feature vector results from projecting the *image-vector* onto the first 40 principal components of PCA.
rows-sum	A 160-dim vectors representing the sums over the rows of the original image and images results from rotating it 9 times.
cols-sum	A 160-dim vectors representing the columns over the rows of the original image and images results from rotating it 9 times.

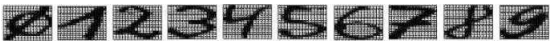

Figure 4. A sample of the handwritten digits data set

pixel. Each sample is represented by five feature vectors as described in Table IV. The Top-Down Approach defined in Algorithm 3 is used to build the class hierarchies.

RBF networks have been used as binary classifiers such that the hidden layer consists of 20 RBFs per class and the number of the input layer nodes equals to the dimension of the feature vector. The classification results are the average of one run of 10-fold cross-validation (CV). First, we construct a tree classifier for each possible view. Table V illustrates the performance of the 5 single-view tree classifiers.

Then, we construct three ensembles: MVF_{single} based on the 5 Single-View tree classifiers (3rd column in Table VI), $MVF(31)$ based on the 31 constructed classifiers (4th column in Table VI) and $MVF(5)$ is constructed by removing similar classifiers from the forest $MVF(31)$ and keeping only the 5 most diverse classifiers using kappa agreement measure. The results show that MVF_{single} outperforms the best Single-View tree classifier and shows better performance than $MVF(31)$. In addition, the top 5 diverse classifiers in $MVF(5)$ are the single-view ones. That is, for digits data sets multi-view trees are similar to single-view ones.

V. CONCLUSIONS

The objective of this study was to investigate multi-view ensembles of tree classifiers using different classifier combination methods. As demonstrated by experiments, Multi-view learning can improve the accuracy in complex pattern recognition problems with a large number of classes. In addition, the trees generated by each individual feature set seem to complement each other by showing part of the discriminating information. This motivates the use of multiple feature sets to generate one consolidated tree, through multi-view hierarchical clustering or k-means clustering. The combination method based on Dempster-Shafer Evidence Theory does not only provide the resulting class encoded by a class label but also provides an estimate of the class memberships of the samples to be classified (soft label). Therefore, a forest can

Table V
RESULTS OF THE FIVE SINGLE-VIEW TREE CLASSIFIERS FOR THE HANDWRITTEN DIGITS

TCM	*image-vector*	*orienthisto*	*pca-40*	*rows-sum*	*cols-sum*
DT	95.8±0.47	96.0±0.59	94.9±0.81	94.0±0.65	93.7±0.95
DS	96.2±0.55	96.5±0.54	95.6±0.61	94.5±0.57	94.0±0.97

Table VI
RESULTS OF THE THREE MULTI-VIEW FORESTS FOR THE DIGITS

TCM	FCM	MVF_{single}	$MVF(31)$	$MVF(5)$
DT	MV	96.8±0.44	94.0±0.64	96.8±0.44
DS	MV	97.1±0.45	94.5±0.61	97.1±0.45
	Min	97.4±0.53	97.4±0.52	97.4±0.53
	Max	97.6±0.54	97.6±0.51	97.6±0.54
	$Mean$	97.6±0.57	95.6±0.63	97.6±0.57
	$Prod$	97.7±0.47	96.5±0.50	97.7±0.47
Best Tree		96.5±0.54	96.6±0.57	96.5±0.54

be constructed not only by majority voting but also by soft combiners such as minimum, maximum, mean, and product. Experiments show that these soft combination methods together with the Dempster-Shafer approach outperform crisp combination methods such as voting.

ACKNOWLEDGMENTS

This work was partially supported by the Transregional Collaborative Research Centre SFB/TRR 62 *Companion-Technology for Cognitive Technical Systems* funded by the German Research Foundation (DFG) and DFG grant SCHW623/4-3.

REFERENCES

[1] L. Kuncheva, *Combining Pattern Classifiers: Methods and Algorithms*. Wiley, 2004.
[2] L. I. Kuncheva and C. J. Whitaker, "Ten measures of diversity in classifier ensembles: Limits for two classifiers," in *IEEE Workshop on Intelligent Sensor Processing*, 2001, p. 110.
[3] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 26, no. 2, p. 123140, 1996.
[4] Y. Freund and R. Schapire, "Experiments with a new boosting algorithm." in *Proc. 13th International Conf. on Machine Learning*, 1996, pp. 148–156.
[5] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.
[6] A. Blum and T. Mitchell, "Combining labeled and unlabeled data with co-training," in *In Annual Conf. on Computational Learning Theory (COLT-98)*, 1998.
[7] A. Gupta and S. Dasgupta, "Hybrid hierarchical clustering: Forming a tree from multiple views," in *Proc. of the Workshop on Learning with Multiple Views, 22nd ICML*, 2005, pp. 35–42.
[8] A. P. Dempster, "A generalization of bayesian inference," *Journal of the Royal Statistical Society*, p. 205247, 1968.
[9] G. Shafer, *A mathematical theory of evidence*. University Press, Princeton, 1976.
[10] E. Mandler and J. Schürmann, "Combining the classification results of independent classifiers based on the dempster-shafer theory of evidence," *Pattern Recognition and Artificial Intelligence (PRAI)*, p. 381393, 1988.
[11] G. Rogova, "Combining the results of several neural network classifiers," *Neural Networks*, vol. 7, no. 5, p. 777781, 1994.
[12] R. Fay, "Feature selection and information fusion in hierarchical neural networks for iterative 3d-object recognition," Ph.D. dissertation, Ulm University, 2007.
[13] F. S. . G. Palm, "Tree structured support vector machines for multi-class pattern recognition," in *Multiple Classifier Systems, MCS2001*, 2001, pp. 409–417.