A Directed Acyclic Graph Based Approach to Multi-Class Ensemble Classification

Esra'a Alshdaifat, Frans Coenen, Keith Dures

Abstract In this paper a novel, ensemble style, classification architecture is proposed as a solution to the multi-class classification problem. The idea is to use a non-rooted Directed Acyclic Graph (DAG) structure which holds a classifier at each node. The potential advantage offered is that a more accurate and reliable classification can be obtained when the classification process is conducted progressively, starting with groups of class labels that are repeatedly refined into smaller groups until individual labels are arrived at. Reported experimental results indicated that the proposed DAG classification model can improve classification performance, in terms of average accuracy and average AUC (Area Under the receiver operating Curve), in the context of some data sets.

1 Introduction

Classification is concerned with the generation of a model, using pre-labelled "training" data, that can be used to allocate labels (classes) to previously unseen data. The nature of the classification problem is characterised by two factors: (i) the number of class labels that can be assigned to an instance (single-label versus multi-label classification), and (ii) the number of classes from which the class labels may be drawn (binary versus multi-class classification). In single-label classification a classifier model is generated using a set of training examples where each example is associated with a single class label c taken from a set of disjoint class labels C ($|C| > 1$). If $|C| = 2$ we have a binary classification problem; if $|C| > 2$, we have a multi-class classification problem. The distinction between single-label and multi-label classification is that in multi-label classification the examples are each associated with a set of class labels Z, $Z \subseteq C$. In the work presented in this paper we focus on the

Esra'a Alshdaifat · Frans Coenen · Keith Dures
University of Liverpool, Department of Computer Science, Ashton Building, Ashton Street, Liverpool L69 3BX, United Kingdom. e-mail: {esraa, coenen, dures}@liv.ac.uk.

multi-class single-label classification problem where examples are associated with exactly one element of the set of class labels C. For simplicity, throughout this work, we will refer to this as "multi-class" classification.

The main issue with multi-class classification is that as the number of classes increases the classification performance tends to degrade [10]. There are three main mechanisms for addressing multi-class classification: (i) using a single all-encompassing classifier, (ii) utilising a collection of binary classifiers such and adopting a One-Versus-All (OVA) [15] or One-Verses-One (OVO) strategy [16], and (iii) using an "ensemble" of classifiers. The ensemble strategy has been shown to work well in the context of the multi-class problem [10, 14, 18]. The ensemble strategy involves using a collection of classifiers typically arranged in either: (i) a "concurrent" form, such as "Bagging" [7]; or (ii) a "sequential" form, such as "Boosting" [9]. In more recent work on ensemble classification, hierarchical arrangements of classifiers have been used [4, 11, 12, 17]. A commonly adopted structure is a binary tree constructed in either a bottom-up or top-down manner [6, 11].

The proposed work is directed at hierarchical ensemble classification where classifiers are arranged in a more sophisticated structure, than a binary tree structure, namely a non-rooted Directed Acyclic Graph (DAG) structure. Nodes at leaves hold classifiers designed to distinguish between individual class labels while the remaining nodes hold classifiers designed to discriminate between groups of class labels. The intuition here is that if we start off with a "coarse grained" classification, moving down to a more "fine grained" classification, a more effective classification model can be produced. This is based on the observation that as the number of classes increases the classification performance tends to degrade as noted above [10]. In this context the advantage offered by the DAG structure is that it enables the inclusion of a greater number of possible class label combinations at each level than in the case of a binary structure. The main challenges associated with this kind of classification are: (i) how to distribute (divide) classes between nodes within the DAG model, (ii) how to determine the starting node among the set of nodes available at the first level in the DAG, and (iii) how to handle the general drawback associated with hierarchical forms of ensemble classification, the "successive miss-classification" problem, whereby if a record is miss-classified early on in the process it will continue to be miss-classified at deeper levels, regardless of the classifications proposed at lower level nodes. To address the first issue a "combination technique" is proposed to assign classes to nodes within the DAG. To address the second challenge the use of Bayesian classifiers is proposed, one per DAG node, which offers the advantage that the probability values produced can be used to determine the best starting first level node. To address the "successive miss-classification" issue two strategies are proposed: (i) following multiple paths within the DAG by utilising the probability values generated by the Naive Bayesian classification model to determine where single or multiple paths should be followed; and (ii) a pruning scheme, applied during the generation process, to eliminate the *weak* classifiers that might affect eventual classification accuracy.

The nature of the proposed multi-class DAG hierarchical ensemble model, and its generation, is fully described in this paper. Its operation is evaluated by compar-

ing with: (i) "stand alone" classification, (ii) Bagging ensemble classification, (iii) One-Versus-One classification, and (iv) a hierarchical binary tree structure based approach. The rest of this paper is organised as follows. Section 2 gives a review of related work on multi-class classification. Section 3 describes the process for generating and operating the proposed DAG ensemble classification model. Section 4 presents an evaluation of the proposed DAG classification model as applied to a range of different data sets. Section 5 concludes the work presented in this paper.

2 Literature Review

This section provides a review of "Ensemble" methods for solving the multi-class classification problem. An ensemble model is a composite model comprised of a number of learners (classifiers), called *base learners* or *weak learners*, that are used together to obtain a better classification performance than can be obtained when using a single "stand alone" model. If the base learners in an ensemble model are all comprised of the same classification algorithm the ensemble model is referred to as an *homogeneous learner*, while when different classification algorithms are used the ensemble model is referred to as *heterogeneous learner* [18]. In general, most ensemble methods are categorised as homogeneous learners [18].

Depending on the relationships between the classifiers forming the ensemble, two main structures can be identified: concurrent (parallel) [7], and sequential (serial) [9]. The hierarchical ensemble methodology is a much more recent approach to solving the multi-class classification problem which involves the generation of a hierarchical "meta-algorithm" [4, 11, 12, 17]. Work to date has been mostly directed at binary tree based hierarchical ensemble classification. Using a binary tree hierarchical classification model the classifiers are arranged in a binary tree formation such that the classifiers at the leaves conduct fine-grained (binary) classifications while the classifiers at non-leaf nodes further up the hierarchy conduct coarse-grained classification directed at categorising records using groups of labels. An example binary tree hierarchy is presented in Fig.1. At the root we distinguish between two groups of class labels $\{a,b,c\}$ and $\{d,e\}$. At the next level we discriminate between smaller groups, and so on, until we reach classifiers that can assign a single class label to the record. The nature of the classifiers held at each node can be of any form.

Although the desired binary tree structure can be constructed in either a bottom-up or a top-down manner, top down construction is the most widely used because it tends to produce a more balanced structure and because it is easier to implement [4]. Using the top down approach the process is as follows: Starting at the root of the tree, a grouping technique is used to segment the records into two clusters, each cluster is labelled with a group-class label. Then, a classifier is trained to classify records using the two group-classes. The process continues recursively until classifiers are arrived at that can assign single class labels to individual records. For classifying a new record a "path" is followed from the root, according to the classification at

each hierarchy node, until a leaf node associated with a single class label is reached. As noted in the introduction to this paper successive miss-classification is a general drawback associated with hierarchical classification, where a miss-classification near the root of the tree is passed on down the hierarchy. To address this problem a multiple path strategy, as previously suggested by the authors in [1, 3], can be adopted whereby multiple paths can be followed within the binary tree hierarchy. However, experiments also reported in [1, 3], indicated that this did not provide an entirely satisfactory solution to the problem.

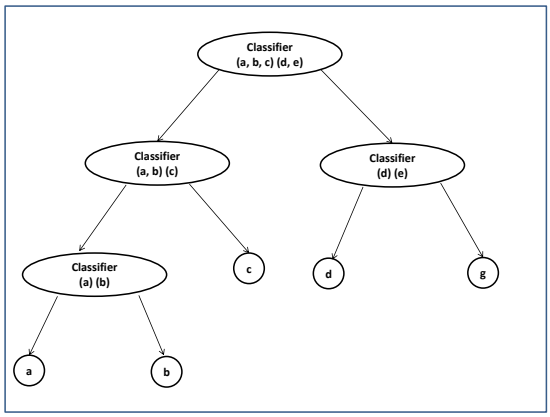

Fig. 1 Binary tree hierarchical classification example.

3 Directed Acyclic Graph (DAG) Classification Model Framework

The nature of the proposed DAG hierarchical classification model is presented in this section. As noted in the introduction to this paper the model is founded on the idea of arranging the classifiers into a hierarchical form by utilising a DAG structure where each node in the DAG holds a classifier. Classifiers at leaves act as binary classifiers while the remaining classifiers (at the beginning and intermediate nodes) are directed at groupings of class labels. An example non-rooted DAG classifier for four class labels $C = \{a, b, c, d\}$, is presented in Fig. 2. In this case, the nodes at the first level are assigned with classes of size three, all possible combination of size $|C| - 1$, while nodes at the second level are assigned with classes of size two ($|C| - 2$). Classifiers at the first level distinguish between three groups of class labels, while classifiers at the leaves discriminate between two individual class labels. The rest of this section is organised as follow. Section 3.1 below explains the generation of the DAG ensemble model in detail. While Section 3.2 presents the operation of the proposed model.

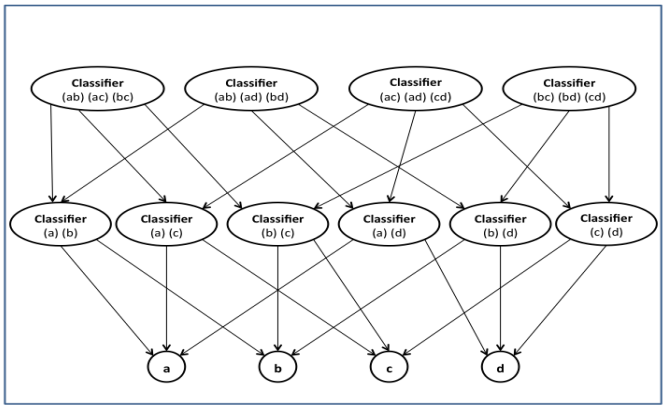

Fig. 2 DAG classification model example.

3.1 DAG Generation

To create the desired DAG model, a classifier needs to be generated for each node in the DAG using an appropriate training set. At "start up" the training set D comprises a set of n records $D = \{r_1, r_2, \ldots r_n\}$ such that each record has a class label associated with it taken from the set C. The process requires a class label grouping mechanism. One way of doing this is to apply a clustering mechanism such a k-means (k-means is particularly well suited because the value of k can be pre-specified). In previous work conducted by the authors [1, 2, 3], in the context of binary hierarchies, it was found that this clustering approach did not work well because similar classes were grouped together early on in the process so that entire branches ended up dealing with very similar classes, ideally we would like individual branches to deal with very different classes so that highly discriminative classifiers can be built at each leaf node. However, identifying such groups is also not straight forward. Instead, the use of a combination mechanism is proposed that covers all potential groupings. The class groupings (sub sets) at each level are determined by finding all possible classes combinations of size $|C| - i$ (where i is the level number, initially $i = 1$). As the process proceeds i is increased by one and consequently the "combination size" is decreased by one. The process continues until the combination size reaches two. The process was used to generate the DAG given in Fig. 2. The number of classifiers to be trained in order to generate the DAG classification model can be calculated using Equation 1.

$$NumberOfClassifiers = 2^N - N - 2 \tag{1}$$

where N is the number of class labels in a given data set.

However, using the above mechanism, the number of classifiers to be generated is large. Breadth and depth pruning mechanisms are thus proposed to reduce the number of classifiers, as well as, to improve the performance of the suggested model. Starting with breadth pruning, in which *weak* classifiers, at each DAG level, are eliminated. The *weak* classifiers are identified by evaluating the classifiers at the

first level and pruning the classifiers associated with an *AUC* value of less than a predefined threshold α. The pruning is transmitted to the remaining levels by only including nodes that are, directly or indirectly, referenced by the first level nodes. Note here that the pruning process at the first level is post pruning, while the pruning at the remanning levels is pre-pruning.

Algorithm 1 presents the proposed DAG generation process with breadth pruning. The input to the algorithm is the training data set D, the set of class labels C, and the breadth pruning threshold α. The DAG is created in a top down manner starting with $k = |C| - 1$ (where k is the combination size) to $k = 2$. Because of the nature of the breadth pruning mechanism the algorithm comprises two procedures: *dagFirstLevelGen*, and *dagNlevelGen*. Starting with *dagFirstLevelGen* procedure, which is responsible for generating and pruning the first level in the DAG model. The process commences (with reference to Algorithm 1) by finding the set of size k class combinations, the set C_k (line 10). We then loop through this set (line 12) and on each iteration: (i) find the set of records D_i that feature the combination $C_i \in C_k$ (line 13); (ii) identify the training and evaluating records, T_i and E_i (lines 14, and 15); (iii) generate a classifier G_i using T_i (line 16); (iv) evaluate the classifier G_i using E_i to produce an *AUC* value (line 17); (v) create a new DAG node, *node*, and add the new node to the set of accumulated level k nodes so far, *NodeSet* (line 18). The next stage is the pruning stage, where the nodes in the *NodeSet* are arranged according to their associated *AUC* values (Line 20). We the loop through this set of ordered nodes (line 21): if the node associated with a particular *AUC* value is less than α (line 22), and its classes are included in the remaining nodes in *NodeSet* (line 23); then the node will be pruned (line 24). (We want to ensure that each class is the subject of at least one classifier.)

After generating and pruning the first level in the DAG model, the next step is to generate the remaining levels. The remaining levels in the DAG are created in a recursive manner using the *dagNlevelGen* procedure. The procedure is invoked with two parameters: (i) the combination size, k, and a reference to the nodes in the current level of the DAG, *CurrentNodes*. For each call to *dagNlevelGen* the set of size k class combinations, the set C_k, is calculated (line 30), then pruning is applied to this set (lines 31-40), where the class combination is only considered if it is a subset of one or more of the previous levels nodes' class sets. We then loop through the pruned combination set (line 42) and on each iteration: (i) find the set of training set records T_i that feature the combination $C_i \in C_k$ (line 43), (ii) generate a classifier G_i using T_i (line 44); (iii) create a new DAG node, *node* (line 45); and (iv) add the new node to the set of accumulated level k nodes so far, *NodeSet* (line 46). We then loop through the set of current nodes (from the previous iteration) and add a link from each current node $CurrentNode_j$ to the new node *node* whenever the set of class labels associated with the new node (C_i) is included in the set of class labels associated with a current node ($C_i \subset CurrentNodes_j.C$). Finally, if k has not yet reached 2, we repeat (line 53).

Because of the flexibility of: (i) the DAG structure and (ii) the combination technique used; it is fairly straightforward (in addition to breadth pruning) to apply depth pruning. Depth pruning in this context is achieved by limiting the number of levels

in the DAG (pre-pruning). Note that the minimum number of levels is always 2. For simplicity, in this paper we considered the generation of the all levels of the DAG model (no depth pruning) and the generation of only 2 levels of the DAG (maximum depth pruning). The idea here is that by applying depth pruning the performance of the suggested model might be improved because the number of classifiers that are required to be learned and evaluated will be decreased; consequently scalability, efficiency, and effectiveness might be improved. For simplicity, and throughout the rest of this paper, we will refer to these approaches as the standard DAG and the two-levels DAG approaches respectively.

3.2 DAG Operation

Section 3.1 above described the process for generating the proposed DAG classification model. After the model has been generated it is ready for use. In this section the operation of the suggested model is explained. As noted earlier, two main challenges are associated with the operation of the proposed DAG model: (i) how to determine the "start node" among the set of nodes available at the first level, and (ii) how to address the general drawback associated with hierarchical forms of ensemble classification, the "successive miss-classification" problem. To address these issues Naive Bayesian probabilities were utilised to determine the best starting node for the DAG model, and also to decide whether a single or a multiple path should be followed at each node; consequently, two strategies are considered for classifying individual records: single path and multiple path.

The single path strategy is the most straight-forward, and involves following a single path through the DAG to classify the record. The classification process commences with the evaluation of all classifiers at the first level and selecting the node who's classifier generates the highest probability value to be the start node, the process continues as directed by the individual node classifications, until a leaf node is arrived at. Leaf nodes, as already noted, hold binary classifiers; thus when a leaf node is reached a binary classification can be conducted and a single class label can be assigned to the record. However, as also already noted, the issue with the single path strategy is that if a mis-classification occurs early on in the process there is no opportunity for addressing this situation later on in the process. The multiple path strategy is designed to handle this problem by allowing more than one path to be followed within the DAG according to a predefined probability threshold σ ($0 \leq \sigma < 1$). In this paper we suggest following up to two branches from each DAG node as a maximum. An alternative strategy might be to follow more than two branches per node, however, this will require more computational resource. Following only two branches also, of course, allows us to make comparisons with the operation of binary tree based hierarchical ensemble classifiers. In cases where more than one path is followed we may end up with a number of alternative class labels at the leaf nodes of the DAG, thus we have a set of "candidate class labels". In order to determine a final classification we take into consideration the Bayesian probability

Algorithm 1 DAG Generation

INPUT: D = The input training data set, C = The set of Classes featured in D,
α = Breadth pruning threshold value
OUTPUT: The generated DAG
Start
$k = |C| - 1$
$dagFirstLevelGeneration(k)$
$dagNlevelGen(k-1, NodeSet)$
End
procedure $dagFirstLevelGen(k)$
    C_k = Set of size k combinations in C
    $NodeSet = \{\}$
    for $i = 1$ to $i = |C_k|$ **do**
        D_i = Set of records in D that feature C_i ($D_i \subset D$)
        T_i= Set of records in D_i for training G_i
        E_i = Set of records in D_i for evaluating G_i
        G_i = Classifier for C_i built using training set T_i
        AUC_i = Evaluation of G_i using E_i
        $NodeSet = NodeSet \cup new\ Node(G_i, C_i, AUC_i)$
    end for
    Arrange nodes in $NodeSet$ in ascending order of associated AUC value
    for $i = 1$ to $i = |NodeSet|$ **do**
        if ($node_i.auc < \alpha$) **then**
            if (classesCovered ($node.C_i$, $NodeSet$)) **then**
                Delete $node_i$
            end if
        end if
    end for
end procedure
procedure $dagNlevelGen(k, CurrentNodes)$
    C_k = Set of size k combinations in C
    for $i = 1$ to $i = |C_k|$ **do**
        for $j = 1$ to $j = |CurrentNodes|$ **do**
            if $C_i \subset CurrentNodes_j.C$ **then**
                $flag$=true, break
            end if
        end for
        if flag == false **then**
            Delete C_i
        end if
    end for
    $NodeSet = \{\}$
    for $i = 1$ to $i = |C_k|$ **do**
        T_i = Set of training records in D that feature C_i ($T_i \subset D$)
        G_i = Classifier for C_i built using training set T_i
        $node = new\ Node(G_i, C_i)$
        $NodeSet = NodeSet \cup node$
        for $j = 1$ to $j = |CurrentNodes|$ **do**
            if $C_i \subset CurrentNodes_j.C$ **then**
                $CurrentNodes_j.childNodes = CurrentNodes_j.childNodes \cup node$
            end if
        end for
    end for
    if $k > 2$ **then**
        $dagNlevelGen(k-1, NodeSet)$
    end if
end procedure

values identified along the paths from each relevant leaf node back to the root node, and produce a set of *accumulated* values. The class with the highest accumulated Bayesian probability value is then selected.

When using the multiple-path strategy, the process commences with the evaluation of the classifiers at the first level and selecting one or two nodes to be the start of the DAG depending on the associated probability (p) for each. This is achieved by considering the two nodes with highest probability values, if both probabilities are greater than a predefined threshold σ then both nodes will be considered to be start nodes, otherwise the node with the highest associated probability value will be considered to be the start node. The process is facilitated as follows. At each DAG node the two class groups associated with the highest probabilities are identified, then if their probabilities are greater than σ both branches will be explored, otherwise the branch with the highest associated p value will be selected. In order to decide the final class label to be assigned to the record among the set of candidate classes, the class associated with the highest accumulated probability value will be selected. The accumulated probability for each candidate class is calculated by summing the probability values identified along the path and then dividing the total by the number of classifiers that were invoked along the path.

4 Experiments and Evaluation

The effectiveness of the suggested DAG classification model was evaluated using twelve different data sets taken from the UCI machine learning repository [5], and pre-processed using the LUCS-KDD-DN software [8]. Ten-fold Cross Validation (TCV) was used throughout. The evaluation measures used were average accuracy and average AUC (Area Under the receiver operating Curve). We will discuss the results in terms of average AUC for simplicity and because of the inclusion of unbalanced data sets within the considered evaluation data sets. For comparison purposes the data sets were also classified using:

1. A "stand alone" **Naive Bayes** classifier [13], the objective being to compare the proposed DAG model with a single conventional model. Naive Bayes was chosen because this was also used in the context of the DAG.
2. A **Bagging** ensemble [7] using Naive Bayes classifiers as the base classifiers, the objective being to compare the proposed DAG model with alternative forms of ensemble.
3. A **Binary Tree Hierarchical Ensemble** classifier of the form described in Section 2, with a Naive Bayes classifier generated for each tree node, and data segmentation to distribute class labels between nodes within the tree, both single-path and multi-path strategies were used, the objective being to compare the proposed DAG model with a simple binary tree model.
4. A **One-Versus-One** (OVO) classification mechanism using support vector machines as the base classifiers [16]. The objective being to compare the proposed

DAG model with a classification mechanism founded on the use of a set of binary classifiers for solving the multi-class classification problems.

The results obtained are presented in the following sections.

4.1 Comparison Between DAG Approaches

This section presents the results obtained using the DAG classification approaches, standard DAG and two-level DAG, coupled with the Single-Path and Multi-Path strategies with respect to the ten different data sets considered in the evaluation. The objective of the comparison is to determine the best DAG approach, and to determine whether following more than one path within the DAG classification model could address the successive miss-classification issue noted earlier. With respect to the Multi-Path strategy a very low threshold of $\sigma = 0.1 \times 10^{-4}$ was used. In previous experiments, not reported in this paper, a range of alternative σ values were considered and it was found that $\sigma = 0.1 \times 10^{-4}$ produced the best performance. The results, in terms of average accuracy and average AUC, are presented in Table 1 (best values highlighted in bold). Considering the two different approaches for the DAG model, standard DAG and two-level DAG, from the table it can be noted that the two-level DAG produced better results than the standard DAG; the suggested reason for this is that two-level DAG is simpler and may therefore feature less overfitting. In addition, it can be observed that using the multi-path strategy can significantly improve the classification accuracy with respect to the single path strategy, especially for the standard DAG approach where the number of levels are higher and consequently the probability of miss-classification is higher.

Table 1 Average Accuracy and AUC values obtained using proposed DAG models coupled with the Single and Multiple path Strategies

Data set	Classes	Single-Path Strategy				Multi-Path Strategy			
		Standard		Two-level		Standard		Two-level	
		Acc.	AUC	Acc.	AUC	Acc.	AUC	Acc.	AUC
Nursery	5	79.83	0.40	**91.44**	0.54	82.32	0.41	90.02	**0.58**
Heart	5	57.01	0.39	**59.91**	**0.40**	56.25	0.37	59.64	**0.40**
PageBlocks	5	91.83	0.53	92.02	0.49	91.87	**0.54**	**92.05**	0.47
Dermatology	6	**87.23**	**0.85**	86.09	0.84	**87.23**	**0.85**	85.51	0.84
Glass	7	69.81	0.46	57.58	0.48	**71.16**	**0.50**	57.18	0.49
Zoo	7	92.18	0.58	**93.18**	**0.59**	92.18	**0.59**	**93.18**	**0.59**
Ecoli	8	**84.43**	**0.41**	82.40	0.40	82.26	0.38	80.89	0.39
Led	10	75.66	**0.76**	**75.75**	**0.76**	75.53	**0.76**	75.66	**0.76**
PenDigits	10	83.59	**0.84**	**83.84**	**0.84**	83.59	**0.84**	**83.84**	**0.84**
Soybean	15	**90.57**	**0.92**	90.04	**0.92**	**90.57**	**0.92**	90.04	**0.92**
Mean		81.21	0.61	81.23	**0.63**	**81.30**	0.62	80.80	**0.63**

4.2 Comparison Between Stand-Alone Classification, Bagging, Binary Tree, OVO SVM and DAG Ensemble Classification

In this section a comparison between the proposed DAG classification model and conventional classification models is presented. In order to conduct a "consistent" comparison between the DAG and existing conventional models, comparison was conducted using the same classifier generator. More specifically, comparison between the operation of a stand alone Naive Bayes classification, Bagging of Naive Bayes classifiers, binary tree hierarchical classification with a Naive Bayes classifier generated for each tree node, and Naive Bayes DAG classification was considered. In addition, a "non-consistent" comparison between the Naive Bayes DAG classification model and OVO SVM was conducted. Recall that the objective of this last comparison was to compare the suggested model with one of the state of the art methods for multi-class classification.

Commencing with the comparison between stand alone classification, Bagging classification, binary tree hierarchical classification, and DAG classification. Table 2 presents the results obtained. Note here that the presented DAG results are the results obtained using the two-level DAG approach coupled with the multiple-path strategy, because the foregoing section established that the two-level DAG multiple path strategy produces a better performance. Because of the lower memory requirements for the two-level DAG compared to the standard DAG, two additional data sets (Chess KRvK, and Letter recognition) were included in addition to those considered in Table 1. From the average AUC given in the table, for each method with respect to the twelve data sets considered, it can be noted that the operation of the proposed DAG classification model is superior to that for stand alone Naive Bayse classification, Bagging ensemble classification, and simple binary tree hierarchical classification where the average AUCs obtained from these methods were 0.59, 0.58, 0.56 respectively, while for the DAG model it was 0.60. The proposed DAG classification model improved the classification accuracy for four data sets (Nursery, Heart, Ecoli, and ChessKRvK), although for one data set (Led) the same result obtained from using stand-alone Naive Bayes classifier.

Comparing with the use of the binary tree based structure a significant improvement was recorded when using the DAG model. The suggested reason for this is that the DAG model provides for greater flexibility than in the case of the binary tree model because of the overlap between class groups represented by nodes at the same level in the hierarchy. The consequence of this is that the overlap partly mitigates against the early miss-classification issue. In addition, pruning the weak classifiers from the DAG model results in a better classification accuracy than in the case of the binary tree structure where all the classifiers are used.

With respect to the comparison with stand alone Naive Bayes classification and Bagging ensemble classification, it is interesting to note that the proposed DAG classification model tends to improve the classification effectiveness with respect to unbalanced datasets such as: Nursery, Heart, Ecoli, and ChessKRvK. It is conjectured that the combination techniques, used to distribute class labels between

Table 2 Average Accuracy and AUC values obtained using "stand-alone" Naive Bayes classification, Bagging, Binary Tree Based hierarchical ensemble classification, and the proposed DAG classification model

Data set	Classes	Naive Bayes Single Model		Bagging Ensemble		Binary Tree Model		DAG Model	
		Acc.	AUC	Acc.	AUC	Acc.	AUC	Acc.	AUC
Nursery	5	**90.22**	0.45	89.96	0.46	89.09	0.58	90.02	**0.58**
Heart	5	54.60	0.34	51.28	0.30	53.77	0.36	**59.64**	**0.40**
PageBlocks	5	**92.69**	**0.52**	92.62	**0.52**	91.27	0.48	92.05	0.47
Dermatology	6	**86.66**	**0.85**	81.00	0.81	84.60	0.84	85.51	0.84
Glass	7	**67.83**	0.49	55.28	0.46	55.28	**0.51**	57.18	0.49
Zoo	7	92.27	0.59	**94.27**	**0.62**	92.18	0.58	93.18	0.59
Ecoli	8	81.70	0.38	**82.56**	**0.39**	64.15	0.27	80.89	**0.39**
Led	10	75.59	**0.76**	75.50	**0.76**	61.13	0.61	**75.66**	**0.76**
PenDigits	10	**84.94**	**0.85**	84.57	**0.85**	81.18	0.81	83.84	0.84
Soybean	15	**91.11**	**0.93**	86.83	0.89	83.71	0.83	90.04	0.92
ChessKRvK	18	**36.32**	0.33	35.66	0.34	33.88	**0.37**	35.36	0.36
LetterRecog	26	**57.37**	**0.57**	56.93	**0.57**	53.44	0.53	55.84	0.56
Mean		**75.94**	0.59	73.87	0.58	70.31	0.56	74.93	**0.60**

nodes within the DAG, helps in the handling of unbalanced datasets. With respect to the "non-consistent" comparison between Naive Bayes DAG and OVO SVM, Table 3 presents the results obtained in terms of average accuracy and average AUC (best results highlighted in bold font). From the table it can be observed that the Naive Bayes DAG produced the best classification accuracy with respect to six of the twelve data sets considered (Heart, Glass, Ecoli, Zoo, Led, and Soybean), although for one data set (Led) the same result was produced using OVO SVM. In the remaining six cases, the OVO SVM produced the best result.

Table 3 Average Accuracy and AUC values obtained using the proposed Naive Bayes DAG coupled with the multi-path strategy, and One-versus-One using SVM as the base classifier

Data set	Naive Bayes DAG		OVO SVM	
	Acc.	AUC	Acc.	AUC
Nursery	90.02	0.58	99.69	**0.64**
Heart	59.91	**0.40**	53.01	0.22
PageBlocks	92.02	0.49	92.58	**0.50**
Dermatology	86.09	0.84	88.73	**0.86**
Glass	57.18	**0.49**	72.04	0.47
Zoo	93.18	**0.59**	94.00	0.58
Ecoli	82.40	**0.40**	82.95	0.36
Led	75.75	**0.76**	75.62	**0.76**
PenDigits	83.84	0.84	98.60	**0.99**
Soybean	90.04	**0.92**	92.54	0.91
ChessKRvK	35.36	0.36	86.40	**0.81**
LetterRecog	55.85	0.56	82.92	**0.83**
Mean	75.14	0.60	84.92	**0.66**

4.3 Note on Efficiency

The number of classifiers required to be generated or evaluated with respect to the suggested DAG approaches can not be determined in advance because of the application of the breadth pruning scheme, but the efficiency can be evaluated according to the generation and classification time. Unfortunately space limitations preclude the presentation of a detailed run time analysis here, however, the analysis of run times indicates that following multiple paths within the hierarchical classification model consumes more run time than in the case of following only a single path, regardless of the adopted structure (binary tree or DAG structure). With respect to the proposed DAG approaches, the two level DAG approach, in which depth and breadth pruning were adopted, requires less run time. Regarding comparison between the DAG structure and binary tree structure, as excepted, the binary tree approach requires less run time because the proposed DAG structure is more complex. Of course, with respect to comparison with the conventional methods, single Naive classifier and Bagging classification, the proposed DAG approach requires more run time. However, the generation of the model needs to be done only once. Regarding the comparison with OVO SVM, the stat-of-the-art approach for multi-class classification, the DAG classification model is more efficient than OVO SVM according to the recorded generation and classification run times.

5 Conclusion

A hierarchical ensemble classification model for multi-class classification using a Directed Acyclic Graph (DAG) structure has been presented. The DAG structure facilitated the use of three mechanisms to address the successive miss-classification problem associated with hierarchical classification. The first mechanism was the combination technique used to group classes across nodes at individual levels in the DAG so that an overlap exists between the class groups; unlike in the case of binary tree based ensemble classifiers where this option is not available. The second mechanism was the option to follow multiple paths down the hierarchy by utilising the probability values generated by the Naive Bayes classifiers generated for each node in the DAG. The third mechanism was the pruning scheme applied to eliminate the weak classifiers that can affect classification effectiveness. The operation of the proposed DAG model was compared with four alternative methods: (i) stand-alone Naive Bayesian classification, (ii) Bagging ensemble classification, (iii) the Binary Tree hierarchical ensemble classifier, and (iv) OVO SVM. From the reported evaluation it was demonstrated that the proposed DAG classification model could be successfully used to classify data in a more effective manner than when alternative conventional methods were used, such as Naive Bayes, Bagging, and OVO SVM with respect to some of the data sets considered in the evaluation. In addition, it was demonstrated that following more than one path in the DAG tended to produce a better classification effectiveness with respect to the majority of the

data sets considered. The evaluation also indicated that the overall performance of the DAG structure was clearly superior to a simple binary tree structure, the most widely adopted structure to generate the hierarchical classification model.

References

1. Alshdaifat, E., Coenen, F., Dures, K.: Hierarchical classification for solving multi-class problems: A new approach using naive bayesian classification. In: H. Motoda, Z. Wu, L. Cao, O.R. Zaïane, M. Yao, W. Wang (eds.) ADMA (1), *Lecture Notes in Computer Science*, vol. 8346, pp. 493–504. Springer (2013)
2. Alshdaifat, E., Coenen, F., Dures, K.: Hierarchical single label classification: An alternative approach. In: M. Bramer, M. Petridis (eds.) SGAI Conf., pp. 39–52. Springer (2013)
3. Alshdaifat, E., Coenen, F., Dures, K.: A multi-path strategy for hierarchical ensemble classification. In: P. Perner (ed.) Machine Learning and Data Mining in Pattern Recognition, *Lecture Notes in Computer Science*, vol. 8556, pp. 198–212. Springer (2014)
4. Athimethphat, M., Lerteerawong, B.: Binary classification tree for multiclass classification with observation-based clustering. In: Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), 2012 9th International Conference on, pp. 1–4 (2012). DOI 10.1109/ECTICon.2012.6254173
5. Bache, K., Lichman, M.: UCI machine learning repository (2013). URL http://archive.ics.uci.edu/ml
6. Beygelzimer, A., Langford, J., Ravikumar, P.: Multiclass classification with filter trees (2007)
7. Breiman, L.: Bagging predictors. Machine Learning **24**(2), 123–140 (1996)
8. Coenen, F.: The LUCS-KDD discretised/normalised arm and carm data library (2003). URL http://www.csc.liv.ac.uk/ frans/KDD/Software/LUCS_KDD_DN
9. Freund, Y., Schapire, R., Abe, N.: A short introduction to boosting. Journal of Japanese Society for Artificial Intelligence **14**(5), 771–80 (1999)
10. Han, J., Kamber, M., Pei, J.: Data Mining: Concepts and Techniques. Morgan Kaufmann (2011)
11. Kumar, S., Ghosh, J., Crawford, M.M.: Hierarchical fusion of multiple classifiers for hyperspectral data analysis. Pattern Anal. Appl. **5**(2), 210–220 (2002). DOI http://dx.doi.org/10.1007/s100440200019
12. Lei, H., Govindaraju, V.: Half-against-half multi-class support vector machines. In: Proc. of the 6th International Workshop on Multiple Classifier Systems, Seaside, CA, USA. Springer (2005)
13. Leonard, T., Hsu, J.S.: Bayesian Methods: An Analysis for Statisticians and Interdisciplinary Researchers. CAMBRIDGE UNIVERSITY PRESS (2001)
14. Oza, N., Tumer, K.: Classifier ensembles: Select real-world applications. Information Fusion **9**(1), 4–20 (2008)
15. Rifkin, R.M., Klautau, A.: In defense of one-vs-all classification. Journal of Machine Learning Research **5**, 101–141 (2004)
16. Tax, D.M.J., Duin, R.P.W.: Using two-class classifiers for multiclass classification. In: ICPR (2), pp. 124–127 (2002)
17. Vural, V., Dy, J.G.: A hierarchical method for multi-class support vector machines. In: Proceedings of the twenty-first international conference on Machine learning, ICML '04, pp. 105–. ACM, New York, NY, USA (2004). DOI 10.1145/1015330.1015427
18. Zhou, Z.H.: Ensemble learning. In: S.Z. Li, A.K. Jain (eds.) Encyclopedia of Biometrics, pp. 270–273. Springer US (2009)