A Multi-Path Strategy for Hierarchical Ensemble Classification

Esra'a Alshdaifat, Frans Coenen, and Keith Dures

University of Liverpool, Department of Computer Science, United Kingdom
{esraa,coenen,dures}@liv.ac.uk.

Abstract. A solution to the multi-class classification problem is proposed founded on the concept of an ensemble of classifiers arranged in a hierarchical binary tree formation. An issue with this solution is that if a miss-classification occurs early on in the process (near the start of the hierarchy) there is no possibility of rectifying this error later on in the process. To address this issue a multi-path strategy is investigated based on the idea of using Classification Association Rule Miners at individual nodes. The conjectured advantage offered is that the confidence values associated with this form of classifier can be used to inform the proposed multi-path strategy. More specifically the confidence values are used to determine, at each node, whether one or two paths should be followed.

Keywords: Hierarchical classification, ensemble classification, multi-class classification, Classification Association Rule Mining

1 Introduction

Single-label classification (as opposed to multi-label classification) is concerned with learning of classifiers, 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. If $|C| = 2$ then we have a simple "binary" classification problem, if $|C| > 2$ it is referred to as a "multi-class" classification problem. In the context of the work described in this paper the focus is on the multi-class single-label classification problem where each example is associated with exactly one element of the class label set C. For simplicity this problem is referred to here as the *multi-class problem*.

An issue with multi-class classification is that when $|C|$ is large the classification accuracy tends to degrade for two reasons: (i) each class is represented by fewer examples in the training dataset than in the case of binary training data, and (ii) a suitable subset of features that can be used to discriminate between large numbers of classes (more than two) is often difficult to identify. One mechanism for addressing this issue is to adopt some ensemble classification technique whereby a set of classification models is used in order to obtain a better composite global model [17]. Much research work has been conducted on ensemble techniques because of the potential benefits of such methods; this research has

demonstrated that it is frequently the case that more accurate and effective classification results can be produced when using ensemble techniques than when using a single model [17, 35, 25, 13]. With reference to the literature, two main mechanisms for arranging the "base classifiers" within an ensemble model can be identified: (i) concurrent (parallel) [23, 8, 9], or (ii) sequential (serial) [15, 27]. A more recent approach for arranging the base classifiers within an ensemble model involves the creation and utilisation of a hierarchy of classifiers [18, 10, 32, 4, 19, 24]. One of the most significant advantages of hierarchical classification is its structural flexibility; the ability to modify or adapt the model so as to fit a particular classification problem.

The work presented in this paper is directed at hierarchical ensemble classification using a binary tree representation where nodes near the root of the tree hold classifiers designed to distinguish between groups of classes while the leaf nodes hold classifiers designed to distinguish between individual class labels. Thus classifiers nearer the root of the hierarchy conduct coarse-grain classification with respect to subsets of the available set of classes while classifiers near the leaves of the hierarchy conduct fine-grain classification. More specifically the work is concerned with the resolution of two issues associated with this form of ensemble classification. The first is how best to distribute (group) classes across the nodes in the hierarchy. The second is concerned with the operation of the hierarchy whereby, if a single-path strategy is adopted, a miss-classification early on in the process (near the root of the hierarchy) cannot be rectified later in the process (further down the hierarchy). Three alternative techniques are proposed in this paper to address the first issue: (i) k-means clustering, (ii) divisive hierarchical clustering, and (iii) data splitting. With respect to the second issue the novel idea presented in this paper is to utilise confidence values generated when using Classification Association Rule Mining (CARM) to inform a proposed multi-path strategy that, in certain circumstances, will cause more than one path to be followed through the hierarchy.

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 the proposed binary tree hierarchical ensemble classification model, while Section 4 describes its operation. Section 5 presents an evaluation of the proposed hierarchical classification mechanism as applied to a range of different data sets. Section 6 summarises the work and indicates some future research directions.

2 Literature Review

This section reviews some of the previous work directed at the multi-class classification problems. It is widely accepted that multi-class problems can be solved in three ways: (i) using "stand-alone" classification algorithms (Section 2.1), (ii) using a "two-class" classifiers (Section 2.2), and (ii) using ensemble classifiers arranged in some specific form (Section 2.3).

2.1 Using Stand-alone Classification Algorithms to Solve Multi-class Classification Problems

Some classification algorithms are specifically designed to handle binary classification, for example support vector machines [31]. While other classification algorithms can be used with respect to any number of class labels; examples of the latter include: decision tree classifiers [26], Classification Association Rule Mining (CARM) [12], Neural Networks [34], k-Nearest Neighbors [6], and Bayesian classification [20]. Among these CARM algorithms are of interest with respect to the work described in this paper. The significance is that CARM incorporates the concept of confidence values which in turn, it is argued in this paper, can be use to determine the most appropriate paths through a hierarchical ensemble classification model. CARM integrates Association Rule Mining (ARM) and classification. CARM algorithms works by applying an association rule mining style algorithm, such as Apriori [1] or FPgrowth [16], to produce classification rules from a training set of previously classified data [12] according to user predefined support (frequency)[1] and confidence[2] thresholds, where the focus is to generate association rules that have only a single class label in the consequent. The generated association rules are referred to as Classification Association Rules (CARs) [21], which collectively form the desired classifier. CARM algorithms can be categorised according to how the pruning of low confidence CARs is performed [12]: (i) two stage or (ii) integrated. In the two stage approach all CARs are generated in the first stage and pruned in the second stage. Examples of this approach include Classification based on Multiple Association Rules (CMAR) [21], and Classification Based on Associations (CBA) [22]. Using integrated algorithms the classifier generation is accomplished in a single processing step encompassing both rule generation and pruning. Examples of this latter approach include Classification based on Predictive Association Rules (CPAR) [33] and Total From Partial Classification (TFPC) [12].

2.2 Using Binary Classifiers to Solve Multi-class Classification Problems

With the availability of many effective binary classification algorithms, the multi-class classification problem can be addressed by utilising a "two-class" classifiers; whereby the multi-class problem is decomposed into a binary classification sub-problems addressed using individual binary classifiers. We can, from the literature, identify three commonly referenced methods of using binary classifiers to solve multi-class problems: (i) One-Versus-All (OVA) [28] (ii) All-Verses-All (AVA) [30] (iii) Error-Correcting-Output-Codes (ECOC) [14]. Among these ECOC has often been demonstrated to outperform "stand-alone" multi-class classification algorithms [14, 29].

[1] The support of a rule describes the number of instances in the training data for which the rule is found to apply [12].

[2] The confidence of the rule is the ratio of its support to the support for its antecedent [12].

2.3 Using Ensemble Classifiers to Solve Multi-class Classification Problems

The use of chains of binary classifiers can be viewed as a form of ensemble classifier. An ensemble classification model is a composite model comprised of a number of classifiers, typically referred to as "base classifiers" or "weak learners", built in order to obtain a better combined global model with more effective classification power than can be acquired from using a single (stand-alone) model. Ensemble methods can be differentiated depending on the relationships between classifiers forming the ensemble, two main types of ensemble can be identified: (i) concurrent ensembles, such as Bagging [23], and (ii) sequential ensembles, such as Boosting [15]. A more recent from of ensemble involves the creation of a hierarchy of classifiers [18, 10, 32, 4, 19, 24]. A common structure adopted for hierarchical classification is a binary tree constructed in either a bottom-up or top-down manner [18, 7]. The top-down model operates in a "divide-and-conquer" manner. The root node contains the entire class label set $\{c_1, c_2, ..., c_n\}$. Starting from the root, the set of class labels at each tree node is recursively divided, and a classifier is trained to discriminate between the two subsets [18]. The process continues until a set of leaf nodes is arrived at. In the bottom-up approach a merging procedure, similar to agglomerative hierarchical clustering, is adopted. At commencement records associate with each class represent the leaf nodes. Nodes separated by the closest "distance" are merged to generate a new node to be associated with a new meta-class [7], and so on until a root node representing the entire class set is arrived at. In this paper an alternative form of generating the desired hierarchies is proposed (see below).

3 Generation of the Hierarchical Model

In this section the generation of the suggested hierarchical model is described. As noted in the introduction to this paper the proposed hierarchical model adopts an ensemble approach founded on the idea of arranging the node classifiers using a binary tree structure. The nature of the classifiers held at each node can be of any form (in previous work the authors have experimented using decision tree and naive bayes classifiers [3, 2]), however, with respect to the work described in this paper (and for reasons that will become apparent later in this paper) classifiers generated using Classification Association Rule Mining (CARM) were used. The intuition behind using ensemble classifiers arranged in a hierarchical form is that it could result in better classification accuracy due to: (i) the established observation that ensemble methods tend to improve classification performance [17, 35, 25, 13], and (ii) smaller subsets of class labels are handled at each hierarchy node thus better results might be produced.

In order to group (divide) the input data D during the hierarchy generation process, three different techniques are considered in this paper: (i) k-means clustering, (ii) divisive hierarchical clustering, and (iii) data splitting. Among these k-means is the most commonly used partitioning method where the records are

divided into k partitions (in our model $k = 2$ was used because of the binary nature of our hierarchies). The divisive hierarchical clustering approach (top-down) was used to generate a hierarchical decomposition of the given input data. The process starts with all examples in one cluster, in each successive iteration, a cluster is divided into smaller clusters until a "best" cluster configuration is achieved (measured using cluster cohesion and separation measures). The data splitting technique comprises a simple "cut" of the data into two groups so that each contains a disjoint subset of the entire set of class labels. The idea behind the use of clustering algorithms is, at each level and branch of the hierarchy, to group the available class labels into two disjoint groups (clusters) so that the classes within each group share some similar characteristics. Note that, using clustering algorithms for dividing the input data, class labels are considered as a "common" attribute.

The process for generating a hierarchical ensemble binary tree is then as follows. Starting at the root of the hierarchy divide the input data into two groups using one of the proposed clustering or splitting approaches (in acknowledgement of the binary nature of our hierarchies, we refer to these groups as the $leftClassGroup$ and the $RightClassGroup$). Then learn a classifier to distinguish between the two class-groups (in this paper we are using a CARM approach). The process continues recursively until we reach node classifiers that can associate single class labels with records.

4 Operation of the Hierarchical Ensemble Classification Model

Section 3 above explained the process for generating the proposed hierarchical model. After the model has been produced it is ready for usage. The most straightforward mechanism with which to classify a new record is to identify a single path leading through the hierarchy, as dictated by the node classifiers, until a leaf node associated with a single class label is arrived at. In this paper we refer to this as the"Most Confident Path" strategy as it operates by selecting the branch associated with the highest confidence value at each node. The process is as follows. Starting at the root of the hierarchy, the node's CARM classifier classifies the new record as belonging to either the $leftClassGroup$ or the $RightClassGroup$ class. This classification then dictates which branch (left or right) that is followed next. The process proceeds in this manner until a single class label is identified (at a leaf node). This will then be the label to be associated with the record.

However, as already noted, a possible issue with the single path strategy is that if a miss-classification occurs early on in the process their is no opportunity for rectifying this situation later on in the process. To address this problem we propose a multiple path strategy whereby more than one path can be followed within the hierarchy according to a predefined confidence threshold σ, $(0 \leq \sigma \leq 100)$. Thus the confidence value $Conf$ associated with class groups ($Conf_{N.leftClassGroup}$ or $Conf_{N.rightClassGroup}$) is used to indicate whether one

or two paths (due to the binary structure of our hierarchy) will be followed according to the σ threshold. If the $Conf$ value of the branch associated with the highest confidence is less than σ both branches emanating from the node will be explored further, otherwise the branch with the highest associated $Conf$ value will be selected.

A further issue that results when more than one path is followed through the hierarchy is that more than one final candidate class label may be arrived at, the question then is which class label to select? Two different approaches are suggested to determining the final resulting class label: (i) best confidence and (ii) best normalised accumulated confidence. Using the best confidence approach the "individual" confidence values associated with the identified candidate classes at the leaf nodes are used to select a final class label. When using the best normalised accumulated confidence approach the "normalised accumulated confidence" values associated with the paths that have been followed are used to select a final class label. Thus we have two variations of the multiple class strategy: (i) multiple path with best confidence class label selection ($MultiPathBestConf$) and (ii) multiple path with best normalised accumulated class label selection ($MultiPathNormalisedConf$).

Starting with the $MultiPathBestConf$ approach, Algorithm 1 presents the suggested procedure. The algorithm is similar to the kind of algorithm that might be used to realise the single path strategy (not included in this paper) except with respect to the use of the σ threshold to decide whether to follow a single path or both paths emanating from a node. At the end of the algorithm a list L, which holds all the identified potential class labels with their associated confidence values for the given case, is processed to select the class label c with the highest confidence value. The procedure $MultiPathBestConf(R, N)$ is called recursively as the process progresses. On each recursion the CARM classifier held at the current node is used to produce a confidence value (line 8), with respect to R for the $leftClassGroup$ and the $rightClassGroup$. We then follow one or two paths according the relative nature of the confidence values returned using the CARM classifier at the current node and the σ threshold. Whenever the size of a class group considered at a node is equal to one (lines 11 and 23), indicating that the group comprises a single class label, the class label and associated confidence value are added to L (lines 12 and 24). At the end of the process (line 35) L is processed to identify the class label with the highest associated confidence value.

With respect to the above it should be recalled that a classifier generated using a CARM algorithm comprises a set of CARs whereby the CARs are typically ordered according to confidence value. CARs with the highest confidence are listed first. If two CARs have the same confidence usually the more general rule (that with the smallest antecedent) will appear first, with more specific rules appearing later[3]. Typically the classifier will also include a default rule to be fired when no other rule fits the given example, which will return the most

[3]Some authors argue that the more specific rule should be listed first, this remains an open question.

Algorithm 1 $MultiPathBestConf$ approach

```
INPUT
R a new unseen record
N a pointer to the current node in the hierarchy (root node at start)
OUTPUT
c the predicted class label of the input record R

L the set of class labels, together with their associated confidence values, maintained as the
procedure progresses, set to {} at start

START PROCEDURE MultiPathBestConf(R, N)
C = Class label set for R with the associated confidence values (Conf) generated using classifier
held at node N (C = {N.leftClassGroup, N.rightClassGroup})
if ( (Conf(N.leftClassGroup) > Conf(N.rightClassGroup) ) then
    if (Conf(N.leftClassGroup) > σ) then
        if (|N.leftClassGroup| == 1) then
            Add class label c_i (c ∈ N.leftClassGroup) to class list L with Conf_{c_i}
        else
            MultiPathBestConf(R, N.leftBranch)
        end if
    else
        MultiPathBestConf(R, N.leftBranch)
        MultiPathBestConf(R, N.rightBranch)
    end if

else
    if (Conf(N.rightClassGroup) > σ) then
        if (|N.rightClassGroup| == 1) then
            Add class label c_i(c ∈ N.rightClassGroup) to class list L with Conf_{c_i}
        else
            MultiPathBestConf(R, N.rightBranch)
        end if
    else
        MultiPathBestConf(R, N.leftBranch)
        MultiPathBestConf(R, N.rightBranch)
    end if

end if
END PROCEDURE MultiPathBestConf(R, N)

Process L and select class label c with highest confidence
```

frequently occurring class label in the original training set. Given a new record to be classified, the first rule whose antecedent matches the record (or the default rule) is used to classify the record. However, our multi-path strategy requires that we have confidence values for both branches emanating from a node. Thus the CARM classifiers used were modified so that, where possible, the confidence values associated with both branches were returned by finding the first rule in the rule base with respect to both classes (where such rules existed).

Algorithm 2 presents the $MultiPathNormalisedConf$ approach. The main difference between the Multiple Path Best Confidence approach and Multiple Path Best Normalised Accumulated Confidence approach is that all confidence values are stored with respect to each path followed (not just the confidence values at the leaf nodes). Consequently a weighting may be derived for each candidate class. We refer to this weighting as a the Normalised Accumulated Confidence ($NormalisedAccumulatedConf$) value. In order to achieve this goal two additional parameters (in addition to the parameters in Algorithm 1) are

Algorithm 2 $MultiPathNormalisedConf$ approach

INPUT
R a new unseen record
N a pointer to the current node in the hierarchy (root node at start)
$AccumConf$ the Accumulated summation of the Confidence Value in the followed path (initially 0.0)
$ConfCount$ Confidence counter keeping the number of confidence values in the followed path
OUTPUT
c the predicted class label of the input record R

L the set of class labels, together with their associated normalised accumulated confidence values, maintained as the procedure progresses, set to {} at start

START PROCEDURE $MultiPathNormalisedConf(R, N, AccumConf, ConfCount)$
C = Class label set for R with the associated confidence values ($Conf$) generated using classifier held at node N ($C = \{N.leftClassGroup, N.rightClassGroup\}$)
if $(Conf(_{N.leftClassGroup}) > Conf(_{N.rightClassGroup}))$ **then**
    $leftAccumConf = Conf(_{N.leftClassGroup}) + AccumConf$
    $leftConfCount = ConfCount + 1$
    if $(Conf(_{N.leftClassGroup}) > \sigma)$ **then**
        if $(|N.leftClassGroup| == 1)$ **then**
            $leftNormalisedAccumConf = leftAccumConf/leftConfCount$
            Add class label c_i ($c \in N.leftClassGroup$) to class list L with the
            $leftNormalisedAccumConf$.
        else
            $MultiPathBestConf(R, N.leftBranch, leftAccumConf, leftConfCount)$
        end if
    else
        if $(Conf(_{N.rightClassGroup})$!= Null) **then**
            $rightAccumConf = Conf(_{N.rightClassGroup}) + AccumConf$
            $rightConfCount = ConfCount + 1$
        else
            $rightAccumConf = AccumConf$
        end if
            $MultiPathBestConf(R, N.rightBranch, rightAccumConf, rightConfCount)$
           $MultiPathBestConf(R, N.leftBranch, leftAccumConf, leftConfCount)$
    end if
else
    $rightAccumConf = Conf(_{N.rightClassGroup}) + AccumConf$
    $rightConfCount = ConfCount + 1$
    if $(Conf(_{N.rightClassGroup}) > \sigma)$ **then**
        if $(|N.rightClassGroup| == 1)$ **then**
            $rightNormalisedAccumConf = rightAccumConf/rightConfCount$
            Add class label c_i ($c \in N.rightClassGroup$) to class list L with the
            $rightNormalisedAccumConf$.
        else
            $MultiPathBestConf(R, N.rightBranch, rightAccumConf, rightConfCount)$
        end if
    else
        if $(Conf(_{N.leftClassGroup})$!= Null) **then**
            $leftAccumConf = Conf(_{N.leftClassGroup}) + AccumConf$
            $leftConfCount = ConfCount + 1$
        else
            $leftAccumConf = AccumConf$
        end if
           $MultiPathBestConf(R, N.rightBranch, rightAccumConf, rightConfCount)$
           $MultiPathBestConf(R, N.leftBranch, leftAccumConf, leftConfCount)$
    end if
end if
END PROCEDURE $MultiPathNormalisedConf(R, N, AccumConf, ConfCount)$

Process L and select class label c with highest normalised accumulated confidence value

used: $AccumConf$ and $ConfCount$, where the $AccumConf$ is used to store the summation of the confidence values for the path followed, while $ConfCount$ is used to store the number of confidence values for the path followed (so that the final accumulated confidence can be normalised). More specifically, the normalised accumulated confidence value, which will be associated with each candidate class label, is calculated as:

$$NormalisedAccumConf = AccumConf \div Confcount \quad (1)$$

where $0 \leq NormalisedConf \leq 100$.

As mentioned above our CARM classifiers were modified to return the confidence values associated with both class labels represented by each node. However, in some cases it was not possible to identify both confidence values. In this case only the single identified confidence value was used when calculating the $NormalisedAccumulatedConfidence$ for a specific path (See Algorithm 2, Lines 23-27 and 44-48).

5 Experimental Evaluation

In this section we present an overview of the experimental set up used to evaluate the proposed binary tree hierarchical ensemble classification model and the results obtained. Fourteen different data sets (with various numbers of class labels) taken from the UCI data repository [5] were used, pre-processed using LUCS-KDD-DN software [11]. Ten-fold Cross Validation (TCV) was adopted throughout. The evaluation measures used were average accuracy and average AUC (Area Under the receiver operating Curve). Both average accuracy and AUC results are presented here, but for simplicity we will discuss the results in terms of average accuracy.

The objectives of the evaluation were as follows:

1. To compare the operation of the three considered class grouping mechanisms: (i) k-means, (ii) divisive hierarchical clustering and (iii) data splitting.
2. To compare the use of the single and multiple path strategies for hierarchical ensemble classification and the two proposed class label selection methods associated with the latter.
3. To compare the operation of the proposed binary tree hierarchical ensemble classification model with stand-alone classification and with alternative established ensemble methods (namely bagging).

The results with respect to the first two objectives are given in Tables 1 and 2. Table 1 gives the TCV classification accuracy values obtained, and Table 2 the AUC results obtained. The results with respect to the third objective are given in Table 3. The results in the context of the above evaluation objectives are discuss in Sub-sections 5.1, 5.2 and 5.3 below.

Table 1. Average Accuracy for the fourteen evaluation datasets using the three different proposed Hierarchical Ensemble Classification strategies.

Data set	Classes	Single Path Stratrgy			Multi. Path Strat. best Conf.			Multi. Path Strat. best Norm. Accum. Conf.		
		K-means	DS	HC	K-means	DS	HC	K-means	DS	HC
WaveForm	3	58.92	57.02	**61.62**	59.02	57.12	61.58	59.02	57.44	61.58
Wine	3	78.99	78.59	**84.28**	78.99	78.59	**84.28**	78.99	78.59	**84.28**
Nursery	5	84.41	**86.81**	53.07	82.53	**86.81**	53.12	80.28	**86.81**	53.06
Heart	5	**53.76**	52.39	53.35	**53.76**	52.39	53.35	**53.76**	52.39	53.35
PageBlocks	5	90.79	89.77	73.73	90.79	89.77	73.88	**91.17**	89.77	73.75
Dermatology	6	74.92	60.28	71.39	75.73	60.28	71.16	**77.34**	60.28	71.16
Glass	7	48.46	**61.56**	52.19	48.46	**61.56**	51.24	48.93	**61.56**	51.71
Zoo	7	**88.00**	85.00	85.00	**88.00**	85.00	85.00	**88.00**	85.00	85.00
Ecoli	8	65.15	**66.57**	52.17	64.55	**66.57**	51.86	64.85	**66.57**	52.17
Led	10	**54.78**	45.53	35.16	52.16	41.38	35.38	52.13	41.53	37.94
PenDigits	10	**61.12**	42.68	50.74	**61.12**	43.89	50.41	60.86	40.12	50.43
Soybean	15	79.01	**88.97**	57.36	79.01	88.43	56.82	79.01	88.43	56.83
ChessKRVK	18	**32.99**	28.13	22.50	32.16	26.58	22.48	31.06	27.28	22.20
LetRecog	26	29.58	**33.12**	21.54	29.17	29.13	21.67	27.87	27.39	21.57
Mean		**64.35**	62.60	55.29	63.96	61.96	55.16	63.81	61.65	55.36

Table 2. Average AUC for the fourteen evaluation datasets using the three different proposed Hierarchical Ensemble Classification strategies.

Data set	Classes	Single Path Stratrgy			Multi. Path Strat. best Conf.			Multi. Path Strat. best Norm. Accum. Conf.		
		K-means	DS	HC	K-means	DS	HC	K-means	DS	HC
WaveForm	3	0.59	0.58	**0.61**	0.59	0.58	**0.61**	0.59	0.58	**0.61**
Wine	3	0.81	0.76	**0.86**	0.81	0.76	**0.86**	0.81	0.76	**0.86**
Nursery	5	**0.43**	**0.43**	0.27	0.42	**0.43**	0.27	0.40	**0.43**	0.27
Heart	5	**0.24**	0.20	0.23	**0.24**	0.20	0.23	**0.24**	0.20	0.23
PageBlocks	5	0.24	0.20	0.25	0.24	0.20	0.23	**0.35**	0.20	0.26
Dermatology	6	0.64	0.46	0.66	0.65	0.46	0.66	**0.68**	0.46	0.66
Glass	7	0.29	**0.31**	0.30	0.29	**0.31**	0.30	0.29	**0.31**	0.30
Zoo	7	**0.52**	0.49	0.49	**0.52**	0.49	0.49	**0.52**	0.49	0.49
Ecoli	8	**0.28**	0.20	0.21	**0.28**	0.20	0.20	**0.28**	0.20	0.21
Led	10	**0.55**	0.45	0.35	0.52	0.41	0.35	0.52	0.41	0.38
PenDigits	10	**0.61**	0.42	0.51	**0.61**	0.43	0.50	0.60	0.40	0.50
Soybean	15	0.76	**0.89**	0.52	0.76	**0.89**	0.52	0.76	**0.89**	0.52
ChessKRVK	18	**0.18**	0.14	0.11	0.17	0.13	0.11	0.17	0.13	0.11
LetRecog	26	0.30	**0.33**	0.22	0.29	0.29	0.22	0.28	0.27	0.22
Mean		**0.46**	0.42	0.40	**0.46**	0.41	0.40	**0.46**	0.41	0.40

5.1 Comparison of Class grouping Mechanisms

As already noted, three techniques were considered for grouping classes at the nodes in the hierarchies: (i) k-means, (ii) data splitting and (iii) divisive hierarchical clustering, indicated by the column headings "K-means", "DS" and "HC" respectively in Tables 1 and 2. Figure 1 shows a comparison between all the suggested strategies in terms of average accuracy. From the figure it can be observed that the best method for class distribution between nodes during the hierarchical model generation was k-means grouping, the second best was data splitting, while hierarchical clustering produced the worst results. According to the results presented in Table 1, it can be seen that by using the k-means clustering algorithm to divide classes between nodes within the hierarchy, a best classification accuracy was obtained for seven of the fourteen datasets considered in the evaluation (Heart, Page Blocks, Dermatology, Zoo, Led, Pen Digits, Chess KRvK). While using data splitting produced a best classification accuracy with respect to five of the datasets (Nursery, Glass, Ecoli, Soybean, Letter Recognition). Using Hierarchical clustering a best classification accuracy was obtained for only two of the datasets considered (Wine, Wave Form).

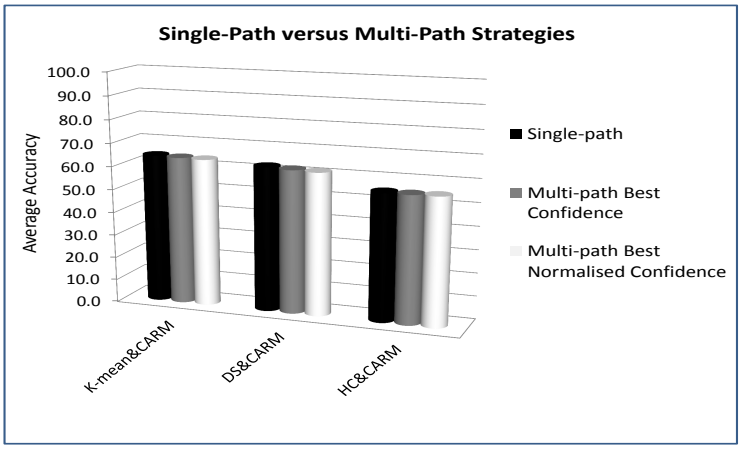

Fig. 1. Comparison between Single-Path and Multi-Path strategies with respect to the different grouping mechanisms (in terms of average accuracy).

5.2 Comparison of Strategies

With respect to the three proposed hierarchical ensemble classification strategies (Single path, Multiple path with best confidence class label selection and Multiple path with best normalised accumulative confidence class label selection) it is interesting to note that their is little difference between the operation of the

strategies. Figure 1 shows three sets of bar charts, one for each strategy, with respect to the mean accuracy values obtained. With reference to the figure it can be seen that the Single-path (the most confident path) strategy produced the best overall result (using k-means grouping). Out of the two multi-path strategies the best confidence approach tended to produce the best performance.

Regarding the Multiple Path strategies a threshold of $\sigma = 70.0$ was used. A range of alternative σ values were evaluated and it was demonstrated that $\sigma = 70.0$ produced the best results.

From the results presented in Table 1 it can be observed that for six datasets (Wine, Nursery, Heart, Glass, Zoo, and Ecoli) the same classification accuracy was obtained regardless of which strategy was adopted. For another six datasets (Wave Form, Led, Pen Digits, Soybean, Chess KRvK, and Letter Recognition) the single path strategy produced the best classification accuracy, although for one dataset (Pen Digits) the same result was produced as for the multi-path best confidence strategy. For two of the datasets (Page Blocks, and Dermatology) the multi-path normalised accumulated confidence strategy produced the best classification accuracy. The reason behind the weakness of the multi-path strategies is that, in some cases, it was not possible to identify both confidence values for a given node branches. Consequently the unknown confidence values affected the results of multi-path strategies tending it to less value than initially anticipated.

Table 3. Accuracy and AUC Comparison between "stand-alone" CARM, a conventional Bagging ensemble, and the best hierarchical ensemble classification strategy from Sub-setion 5.2

Data set	Classes	CARM		Bagging		Best Hier. Esemb. Class.	
		Acc.	AUC	Acc.	AUC	Acc.	AUC
WaveForm	3	60.04	0.60	60.76	**0.61**	**61.62**	**0.61**
Wine	3	71.88	0.74	61.48	0.61	**84.28**	**0.86**
Nursery	5	73.94	0.36	73.94	0.36	**86.81**	**0.43**
Heart	5	51.70	0.20	45.49	0.24	**53.76**	**0.24**
PageBlocks	5	89.99	0.21	89.95	0.21	**91.17**	**0.35**
Dermatology	6	77.00	0.66	72.12	0.62	**77.34**	**0.68**
Glass	7	**65.05**	**0.43**	53.30	0.31	61.56	0.31
Zoo	7	**94.00**	**0.59**	83.00	0.46	88.00	0.52
Ecoli	8	49.98	0.12	37.90	0.07	**66.57**	**0.28**
Led	10	**67.28**	**0.67**	67.09	**0.67**	54.78	0.55
PenDigits	10	75.99	0.76	**77.09**	**0.77**	61.12	0.61
Soybean	15	84.01	0.86	73.15	0.77	**88.97**	**0.89**
ChessKRVK	18	17.64	0.07	17.43	0.06	**32.99**	**0.18**
LetRecog	26	30.91	0.31	30.72	0.31	**33.12**	**0.33**
Mean		64.96	0.47	60.24	0.43	**67.11**	**0.49**

5.3 Comparison of Hierarchical Ensemble Classification with Conventional approaches

Table 3 presents the results obtained using a "stand alone" CARM and Bagging together with the best results from Tables 1 and 2. The proposed hierarchical ensemble classification was compared with CARM because we wanted to compare the operation of the proposed ensemble approach with the operation of the more traditional stand-alone form of classification; CARM was used for this purpose because the classifiers held at the individual nodes in our tree ensembles were also generated in this manner (clearly we could also have compared with alternative forms of stand alone classification). We compared with bagging because we also wanted to compare the operation of our hierarchical ensemble classification with an alternative form of ensemble. From Table 3 it can be seen that the proposed hierarchical techniques can significantly improve the classification accuracy with respect to ten of the fourteen datasets considered (Wave Form, Wine, Nursery, Heart, Page Blocks, Dermatology, Ecoli, Soybean, Chess KRvK, and Letter Recognition). In the remaining four cases, the stand-alone CARM classifier produced the best result for three of the datasets (Glass, Zoo, and Led), while the Bagging ensemble classifier produced the best result for only one dataset (Pen Digits).

6 Conclusion and Future Work

In this paper hierarchical classification using a binary tree structure, for solving multi-class problems, has been considered. To generate such a hierarchical model three different techniques to distribute class labels between nodes within the hierarchy were proposed: (i) k-means, (ii) data splitting, and (iii) hierarchical clustering. CARM classifiers were used at each hierarchical node. By utilising the confidence values generated by CARM classifiers two approaches were proposed: (i) Single-Path, and (ii) Multi-Path. The later one coupled with two alternatives to determining the final resulting class label: Best Confidence and Best Normalised Accumulated Confidence. The conjecture here was that a criticism of hierarchical classification ensembles is that a miss-classification made early on in the hierarchy cannot be rectified later on and that this might hinder the effectiveness of the operation of such classifiers.

From the reported experimental results, presented in this paper, it was demonstrated that the proposed hierarchical classification model tends to perform better than stand-alone classifiers and other forms of ensemble classifier such as bagging, with respect to some datasets considered in the evaluation; especially data sets that featured a large number of class labels.

Although best overall result was obtained using the proposed Single-path strategy (the most confident path) the Multi-path strategy coupled with either best confidence or best normalised accumulated confidence improved the classification accuracy with respect to some datasets considered in the evaluation. An issue with Multiple Path strategies is that the confidence values that were used to determine wether one or two paths emanating from a node will be followed,

can not be always obtained. Consequently, in many cases, it was not possible to follow more than one path even if this might have been desirable.

With respect to future work the authors intend to investigate alternative forms of hierarchical ensembles using more sophisticated structures than the binary tree structures considered in this paper, such as Directed Acyclic Graph (DAG). The idea is that by using DAGs many paths can be followed because of the many possible combinations of class labels at each level.

References

1. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: VLDB94. pp. 487–499 (1994)
2. Alshdaifat, E., Coenen, F., Dures, K.: Hierarchical classification for solving multi-class problems: A new approach using naive bayesian classification. In: Motoda, H., Wu, Z., Cao, L., Zaïane, O.R., Yao, M., Wang, W. (eds.) ADMA (1). Lecture Notes in Computer Science, vol. 8346, pp. 493–504. Springer (2013)
3. Alshdaifat, E., Coenen, F., Dures, K.: Hierarchical single label classification: An alternative approach. In: Bramer, M., Petridis, M. (eds.) SGAI Conf. pp. 39–52. Springer (2013)
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)
5. Bache, K., Lichman, M.: UCI machine learning repository (2013), `http://archive.ics.uci.edu/ml`
6. Bay, S.D.: Combining nearest neighbor classifiers through multiple feature subsets. In: Proc. 17th Intl. Conf. on Machine Learning. pp. 37–45. Morgan Kaufmann (1998)
7. Beygelzimer, A., Langford, J., Ravikumar, P.: Multiclass Classification with Filter Trees (Jun 2007), `http://hunch.net/\~{}jl/projects/reductions/mc\_to\_b/invertedTree.pdf`
8. Breiman, L.: Bagging predictors. Machine Learning 24(2), 123–140 (1996)
9. Breiman, L.: Random forests. In: Machine Learning. pp. 5–32 (2001)
10. Chen, Y., Crawford, M.M., Ghosh, J.: Integrating support vector machines in a hierarchical output decomposition framework. In: In 2004 International Geosci. and Remote Sens. Symposium. pp. 949–953 (2004)
11. Coenen, F.: The LUCS-KDD discretised/normalised arm and carm data library (2003), `http://www.csc.liv.ac.uk/~frans/KDD/Software/LUCS_KDD_DN`
12. Coenen, F., Leng, P.: The effect of threshold values on association rule based classification accuracy. Journal of Data and Knowledge Engineering 60(2), pp345–360 (2007)
13. Dietterich, T.G.: Ensemble methods in machine learning. In: Proceedings of the First International Workshop on Multiple Classifier Systems. pp. 1–15. MCS '00, Springer-Verlag, London, UK, UK (2000), `http://dl.acm.org/citation.cfm?id=648054.743935`
14. Dietterich, T.G., Bakiri, G.: Solving multiclass learning problems via error-correcting output codes. JAIR (1995)
15. Freund, Y., Schapire, R., Abe, N.: A short introduction to boosting. Journal of Japanese Society for Artificial Intelligence 14(5), 771–80 (1999)

16. Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. SIGMOD Rec. 29(2), 1–12 (May 2000), http://doi.acm.org/10.1145/335191.335372
17. Jiawei, H., Micheline, K., Jian, P.: Data Mining: Concepts and Techniques. Morgan Kaufmann (2011)
18. Kumar, S., Ghosh, J., Crawford, M.M.: Hierarchical fusion of multiple classifiers for hyperspectral data analysis. Pattern Anal. Appl. 5(2), 210–220 (2002)
19. 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)
20. Leonard, T., Hsu, J.S.: Bayesian Methods: An Analysis for Statisticians and Interdisciplinary Researchers. CAMBRIDGE UNIVERSITY PRESS (2001)
21. Li, W., Han, J., Pei, J.: Cmar: Accurate and efficient classification based on multiple class-association rules. In: Proceedings of the 2001 IEEE International Conference on Data Mining. pp. 369–376. ICDM '01, IEEE Computer Society, Washington, DC, USA (2001), http://dl.acm.org/citation.cfm?id=645496.657866
22. Liu, B., Hsu, W., Ma, Y.: Integrating classification and association rule mining. In: Proc. KDD-98 conference (AAAI 1998). pp. 80–86 (1998)
23. Machov, K., Bark, F., Bednr, P.: A bagging method using decision trees in the role of base classifiers (2006)
24. Madzarov, G., Gjorgjevikj, D., Chorbev, I.: A multi-class svm classifier utilizing binary decision tree (2008)
25. Oza, N., Tumer, K.: Classifier ensembles: Select real-world applications. Information Fusion 9(1), 4–20 (Jan 2008), http://dx.doi.org/10.1016/j.inffus.2007.07.002
26. Quinlan, J.R.: Induction of decision trees. Machine Learning 1(1), 81–106 (1986)
27. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann (1993)
28. Rifkin, R.M., Klautau, A.: In defense of one-vs-all classification. Journal of Machine Learning Research 5, 101–141 (2004)
29. Schapire, R.E.: Using output codes to boost multiclass learning problems. In: Machine Learning: proceedings of the Fourteenth International Conference , 1997 (ICML-97) (1997)
30. Tax, D.M.J., Duin, R.P.W.: Using two-class classifiers for multiclass classification. In: ICPR (2). pp. 124–127 (2002)
31. Vapnik, V.N.: The Nature of Statistical Learning Theory. Statistics for Engineering and Information Science, Springer (2000)
32. 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. pp. 105–. ICML '04, ACM, New York, NY, USA (2004), http://doi.acm.org/10.1145/1015330.1015427
33. Yin, X., Han, J.: Cpar: Classification based on predictive association rules (2003)
34. Zhang, G.P.: Neural networks for classification: A survey. IEEE Transactions on Systems, Man, and Cybernetics-Part C: Applications and Reviews 30(4), pp451–462 (2000)
35. Zhou, Z.H.: Ensemble learning. In: Li, S.Z., Jain, A.K. (eds.) Encyclopedia of Biometrics, pp. 270–273. Springer US (2009), http://dblp.uni-trier.de/db/reference/bio/e.html#Zhou09